A Comparative Study of Statistical Models for Feature Selection Methods in Text Categorization

Tansel Sarıhan

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
September 2019
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Cem Ergün
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Adnan Acan

2. Asst. Prof. Dr. Cem Ergün

3. Asst. Prof. Dr. Mehtap Köse Ulukök

ABSTRACT

In consequence of change and developments in the world of technology, the data have been started to transfer to the digital environment rapidly and categorization task of digital documents has become difficult and complicated. Therefore, researchers have focused on doing more research in field of machine learning to provide a more effective solution in terms of resources and time. A major problem of text categorization is the high dimension of the feature space. Feature selection methods are widely used for choosing the subset of features in last decades. In order to maximize the text classification efficiency, some machine learning algorithms and feature selection methods are studied in a comparative way. The experiments are conducted with Reuters-21578 "ApteMod" version, The 4-Universities and 20-Newsgroups "bydate" version datasets. Many topics are discussed from gathering data to organizing data with diffent preprocessing and term weighting approaches to perform test by using the feature selection methods and many classification algorithms. The idea behind of feature selection is that determining of the importance of words that are discriminative for categorization task and removing the non-informative terms. In this regard, CHI-Square, Mutual Information, Galavotti-Sebastiani-Simi Coefficient and Document Frequency metrics are studied for feature selection process. The TF-IDF and probability-based term weighting approaches are used to prepare the texts for classification process. Then to get the best achievement for the classifiers and feature selection methods, the effectiveness of system is evaluated with performance evaluation metrics such as accuracy score, precision, recall and *f*-measure.

Keywords: Feature Selection Methods, Text Categorization, Term Weighting Performance Evaluation

ÖZ

Teknoloji dünyasındaki değişim ve gelişmelerin sonucunda, veriler hızla dijital ortama aktarılmaya başlanmış ve böylece, dijital belgelerin sınıflandırılması zor ve karmaşık hale gelmiştir. Bu sebepten dolayı araştırmacılar bu probleme zaman ve kaynak kullanımı açısından daha verimli bir çözüm sağlamak için makine öğrenmesi alanında daha fazla araştırma yapmaya odaklanmıştır. Metin sınıflandırmanın ana sorunu, özellik alanının yüksek boyutudur. Özellik seçim yöntemleri, son yıllarda özelliklerin alt kümesini seçmek için yaygın olarak kullanılır. Metin sınıflandırma verimliliğini en üst düzeye çıkarmak için, bazı makine öğrenme algoritmaları ve özellik seçimi yöntemleri karşılaştırmalı olarak incelenmiştir. Deneyler Reuters-21578 "ApteMod", The 4-Universities ve 20-Newsgroups "bydate" verisetleri ile gerçekleştirilmiştir. Özellik seçim yöntemlerini ve birçok sınıflandırma algoritmasını kullanarak farklı metin ön işleme ve terim ağırlıklandırma yaklaşımlarına kadar birçok konu tartışılmaktadır. Özellik seçiminin arkasındaki fikir, metinlerin kategorilerini ayırabilecek nitelikte olan kelimelerin öneminin belirlenmesi ve bilgilendirici olmayan terimlerin kaldırılmasıdır. Bu bağlamda, özellik seçimi için Ki-kare, Karşılıklı bilgi, Galavotti-Sebastiani-Simi Katsayısı ve Döküman frekansı ölçümleri incelenmiştir. TF-IDF ve olasılık temelli terim ağırlıklandırma yaklaşımları, metinleri sınıflandırma sürecine hazırlamak için kullanılmıştır. Daha sonra en iyi sınıflandırıcıları ve özellik seçim metriklerini elde etmek için, sistemin etkinliği accuracy, precision, recall ve *f*-ölçüsü gibi performans değerlendirme ölçütleri ile değerlendirilmiştir.

Anahtar Kelimeler: Özellik Seçim Yöntemleri, Metin Sınıflandırma, Terim Ağırlıklandırma, Performans Değerlendirme

ACKNOWLEDGMENT

I would like to truly thank my thesis supervisor Asst. Prof. Dr. Cem Ergün for his support and guidance during my studies. I would also like to thank Asst. Prof. Dr. Abdülkadir Görür for his valuable advice and support in my studies. I would like to thank Assoc. Prof. Dr. Adnan Acan, Assoc. Prof. Dr. Gürcü Öz and Asst. Prof. Dr. Mehtap Köse Ulukök as well for serving as jury members.

I would like to thank my friends Mahmut Sevince, Emre Yıldız, Ömer Demir, İmren Toprak, Özgü Taçyıldız, Begüm Koru and Selin Bitirim for sharing their experiences with me and for their help.

Finally, I must express my very profound thanks to my family and to my girlfriend Simge Saygı for providing me endless support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BoW	Bag of Words
C	Category
CHI	CHI-Square
CM	Confusion Matrix
CR	Classification Report
D	Document
DF	Document Frequency
DT	Decision Tree
FN	False Negative
FP	False Positive
GNB	Gaussian Naïve Bayes
GSS	Galavotti-Sebastiani-Simi Coefficient
IDF	Inverse Document Frequency
IG	Information Gain
kNN	k-Nearest Neighborhood
LLSF	Linear Least Square Fit
MI	Mutual Information
ML	Machine Learning
MLP	Multilayer Perceptron
MNB	Multinomial Naïve Bayes
NB	Naïve Bayes
NLP	Natural Language Processing
NLTK	Natural Language Toolkit

NN	Neural Network
OVR	One-vs-Rest
POSTAG	Part-of-Speech Tagging
RBF	Radial Basis Function
RCV1	Reuters Corpus Volume-1
RF	Random Forest
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TC	Text Categorization
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
TN	True Negative
TP	True Positive
TS	Term Strength
VSM	Vector Space Model
WebKB	Web Knowledgebase
X^2	CHI-Square

Chapter 1

INTRODUCTION

In consequence of change and developments in the world of technology, documents which are in the printed form such as news, books, e-mails and scientific articles, have started to be transferred to digital environment in rapidly. The categorization task of digital documents has become difficult and complicated hence, there is a need for people to access the desired information more organized and quickly from among many such sources [1]. Experts who divide the materials into classes and categories, need to spend a lot of time and effort due to the rapid increase of information in the electronic environment.

Traditional text classification methods, which depend on human labor, have become inadequate against the speed of developing the technology. Therefore, researchers have focused on doing more research in the field of automated text classification and machine learning (ML) to provide a more effective and efficient solution in terms of resources and time [1] [2].

The main idea behind of text classification is to assign previously defined categories to text documents by using accumulated information during the learning stage of classification algorithms [3]. Once the text data obtained, it is necessary to perform feature extraction from the data to put it into a suitable form for the categorization task [4]. Since text data contains many words by nature, the high dimensionality of the

feature space can become a significant problem. Hence, text classification will require additional computation and become more expensive in terms of time and hardware [5]. Besides that, performance will be adversely affected as there will also be noisy attributes in the high dimensional attribute space. To solve such problems related to high dimensional and noisy data, feature selection plays an essential role in text categorization tasks. In general, feature selection is choosing the subset of features that can be helpful to represent the documents among all features in the dataset. Therefore, better performance can be achieved by using fewer resources [6].

In this thesis, we examined some important feature selection methods such as CHI-Square (CHI) [7], Mutual Information (MI) [8], Galavotti Sebastiani Simi (GSS) Coefficient [9] [10], and used several algorithms such as Naive Bayes (NB) [11], Support Vector Machines (SVMs) [12], k Nearest Neighborhood (kNN) [13]. Experiments are conducted with Reuters-21578 [14], The 4-Universities [15], 20-Newsgroups [16] datasets.

1.1 Literature Review

In this section, a literature survey on text categorization is presented, including subtopics such as preprocessing techniques, document representation, and term weighting approaches and feature selection methods. Besides, these subfields are briefly mentioned.

1.1.1 Text Categorization (TC)

Text categorization is a challenging research subject with the need to classify and organize electronic documents, which are increasing in number. Until now, it has been successfully applied in many areas such as grouping news, sorting scientific articles by their topics, classifying books and similar materials by authors, and detection of

spam mails. Categorization is the task of assigning the documents to the predefined categories by extracting the characteristics from the words with statistical analysis.

To automatically classify text data, we can use the fast computing power of the recent processors and their reliable processing ability to do more efficient work in less time. Although manually classifying text documents is used as an approach, performing this process on a high number of documents requires much human labor and it is not possible to categorize all documents. Therefore, the need for automated TC systems is increasing day by day tremendously [45].

Hayes, P. J., Weinstein, S. P. [17] proposed a content-based indexing system for news stories. In the beginning, Boolean keyword technique is used for indexing. Since the binary keyword approach could not meet the need for high accuracy, a rule-based approach is adopted. After defining rule modules for concept recognition and categorization processes, they combined these rules and inferred the correct category among similar categories for news.

Stamatatos E., Fakotakis N., Kokkinakis G. [18] described an approach to categorize the texts automatically by their genre and authors. They carried out their studies using stylometric methods which relies on NLP tools. Since each author has a different style, they examined the elements such as word groups used in sentences, usage style of punctuations, number of words in the sentence, vocabulary richness, and classified the texts according to the authors and types. The Modern Greek Corpora is used for experiments, and they achieved better results than lexically based methods for categorization task.

Goudjil M., Koudil M., Hammami N., Bedda M., Alruily M. and Smar Q. [19] proposed a batch-mode learning method for classifying Arabic texts to the support vector machines.

Sahami M., Dumais S., Heckerman D., Horvitz E. [20] introduced a Bayesian approach for filtering junk e-mails. Domain-specific words and domain types of the sender are used instead of the full content of e-mails to label them as spam or not. They used Zipf's Law for eliminating less frequent features, and mutual information is used to select sub-feature set. Finally, it is figured out that the proposed method is successful at eliminating 80% of spam mails.

1.1.2 Preprocessing

As already mentioned, text categorization is the task of organizing text documents under one or more categories according to their contents. Preprocessing methods should be used in order to put the texts into a much proper structure for the classification process. Briefly, the text preprocessing consists of a series of linguistic processes, such as splitting words from spaces, tokenizing, case normalization, removing punctuation and numerical characters, eliminating stop words, and reducing words to their root forms. By using text preprocessing techniques, both raw text data will become more suitable for categorization process and high dimensionality will be reduced.

Srividhya V., Anitha R. [21] did a study to evaluate the effectiveness of preprocessing techniques on text categorization results. They used the Reuters 21578 dataset and the SMART stop word list for their work. First, the words which are frequently used in English language such as "a", "in", "an", "is", "are", were eliminated from documents. Porter stemmer algorithm is used to reduce all words to their root form. Vector space

model (VSM) is used to represent documents and term frequency–inverse document frequency (TF-IDF) is used as a term weighting method, then a threshold is obtained to filter terms according to their document frequency. If document frequency is less than the threshold, the term is removed from vocabulary. As a result, they observed that although all methods have a positive impact on performance individually, combining them leads to better performance.

Leopold E., Kindermann J. [22] did a study with support vector machines, using some text preprocessing and document representation techniques. Reuters-21578, "Die Tageszeitung" and "Frankfurter" collections are used in the study. The methods of lemmatization and document representation are examined. Since preprocessing is easy for the English language, the study also included collections in the German language. Experiments are conducted with five different linguistic preprocessing techniques in combination and according to the results of the experiments, the usage of lemmatization is slightly improved the categorization results. The experimental results also indicated that when the SVMs are used, it is seen that no need to apply preprocessing methods to textual data.

Albishre K., Albathan M., Li Y. [23] studied the role of text cleaning on 20 Newsgroups dataset. Since WebKB database consists of e-mail texts, they kept only subject field and text body of the contents. After applying text preprocessing methods such as removal of stopwords, stemming, size of the dataset is reduced to 21.8MB from 61.6MB. When the original and preprocessed dataset are evaluated, it is seen that there is no loss in performance.

Kadhim A. I., Cheah Y. N., and Ahamed N. H. [24] implemented TF-IDF and Singular Value Decomposition (SVD) to reduce the dimensionality using k-means algorithm for document clustering. In the study, BBC News and BBC sports datasets are used to simulate results. According to their experimental results, the vector dimension is reduced from 9636 features to 100 features and classification accuracy remained approximately the same.

Chandrasekar P., Qian K. [25] studied on impact of preprocessing to evaluate the performance of NB classifier. The Enron's dataset which consists of 6000 e-mails. The methods such as removing noisy words, feature reduction, stemming are applied. The comparisons between the results of both preprocessed and non-preprocessed data indicate that preprocessing task can improve the accuracy of a classification and reduce the numbers of false positives.

Isa D., Lee L. H., Kallimani V. P., Rajkumar R. [26] proposed a hybrid approach to preprocess and classify the e-mails. Experiments are performed on 20-Newsgroups dataset. They implemented NB classifier as a document vectorizer and SVM is used to classify the data. According to the experimental results of their study, the proposed hybrid method is better than the traditional method.

1.1.3 Document Representation and Term Weighting

While the human brain has a very high capacity to understand and express complex concepts, computers cannot understand such complex concepts directly. In other words, when a person reads a sentence and understands it, such a sentence is a series of signs or digital data for the computer. It is necessary to put the text documents into an appropriate form because computers cannot interpret text data as humans do.

Since text documents are not numerical data by nature, it is necessary to express the documents to be categorized in a vector space after preprocessing techniques are applied.

The vector space model proposed by Salton G., Wong A. and Yang C. S. in 1975 is commonly used to represent documents numerically [27]. According to the VSM, documents and words are expressed as a matrix of *N*x*M* size where *N* is the number of documents and, *M* is the number of words. This representation method is also known as Bag of Words (BoW) model [28].

Term weighting methods are developed to transform the terms into numerical values. Some of commonly used such methods are Boolean weighting, document frequency (DF) weighting, TF-IDF methods [29] [30] [31].

Liu C. Z., Sheng Y., Wei Z., Yang Y.Q. [32] did a research based on improved TF-IDF algorithm for text classification. They used a dataset which is provided by Fudan University Laboratory and Sogou Lab. Word2Vec [33] is used to train the text and creating the text vectors. According to the results obtained from experiments, improved TF-IDF and Word2Vec trained text classification significantly improved in term of *F*-measure.

Liu Y., Loh H. T., Sun A. [34] introduced a new term weighting approach to handle imbalanced data in text classification. They proposed a probabilistic weighting approach. Benchmark is performed on MCV1 and Reuters-21578 datasets. The proposed method replaces the IDF part with a probability of terms in standard TF-IDF scheme. SVMs and Complement NB is used as classifiers. When the results are

compared, the proposed method improved the macro–averaged *F* values on both datasets.

Özgür A., Özgür L., Güngör T. [35] did a study on comparison of TF-IDF and Boolean weighting methods with SVMs on Reuters-21578 dataset. According to the experimental results, it is found that TF-IDF weighting method outperformed the Boolean weighting method.

1.1.4 Feature Selection

Feature selection task is the process of selecting the best of all features. The primary purpose under feature selection is to select the smallest subset of features that can represent the original data. According to the Ladha L. and Deepa T. [36], increasing the speed of algorithms and reducing resource requirements such as storage, eliminating meaningless and noisy information, reducing runtime, improving quality and performance are the advantages of feature selection.

Rogati M., Yang Y. [37] proposed a high-performance feature selection method for TC, which consists of CHI combined with DF or information gain (IG). Since a large data set such as RCV1 is used in their study, they tend to use the filtering approach. Among the feature selection methods, the CHI is the best method which is independent of the classifier algorithm. It is beneficial to remove features with low document frequency.

Hawashin B., Mansour A. M., Aljawerneh S. [38] proposed CHI based feature selection method for Arabic language in TC. They compared their proposed method with various feature selection methods on Akhbar Alkhalij and Alwatan datasets.

According to results, the proposed method has better performance than IG, pure CHI, TF-IDF and DF.

Yang Y., Pedersen J.O. [6] studied on five different feature selection metrics. DF, CHI, MI, IG and Term Strength (TS) metrics are compared on Ohsumed and Reuters-21578 datasets by using kNN and LLSF classifiers. Results indicate that X^2 test and IG are most successful metrics among such five feature selection methods.

Saleh S. N., El-Sonbaty Y. [39] focused on to remove redundant features and selecting the best relevant features for the categories according to MI. The Reuters-21578 dataset is used for benchmarking purpose in the study. SVMs and NB are used as classifiers. When the results compared with the study of Bakus J., Kamel M. S. [40], it is found that the proposed algorithm better than MIFS-C and IG measures.

Zhang H., Ren Y., Yang X. [41] performed a study on IG algorithm and enhanced the performance of this algorithm. The Chinese text corpus from FuDan University is used in this study and since used corpus is imbalanced, the improvement of IG was necessary. According to the experiments made by comparisons, a successful performance has observed.

1.1.5 Classification Algorithms

Various methods such as SVM, kNN, decision trees (DT), neural networks (NNs), and regression methods are used in the TC field to classify text documents.

Sebastiani F. [12] did a survey to discuss leading approaches for those in TC and ML areas. According to the survey performed by Sebastiani F., SVMs, sample-based and regression methods exhibit the best results in TC. Also, SVMs, sample-based methods

and regression methods exhibit the best performance and NNs work well on textual data. Probabilistic classification algorithms such as NB and Rocchio [42] are seen as the worst for text data. DTs are seen as the closest method to SVMs when compared to SVMs. The worst performing classifier is WORD, a word-based classification implemented by Yang Y. [43] and it is not included in any learning process.

Joachims T. [42] did a study to explore the effectiveness of SVMs on TC. Since SVMs are not affected by high input space and do not require techniques such as text preprocessing, it is found to facilitate TC.

Aggarwal C. C., Zhai C. X. [44] surveyed for TC and classifiers. The pros and cons aspects of classification algorithms are presented as a table and also feature selection methods such as IG and CHI are examined. According to observations, the best performance is obtained by SVM, kNN and NNs, NB methods, respectively.

1.2 Motivation and Objectives

Text Categorization is a vital intelligence information processing technology. This technology has high value in information retrieval, electronic governments, information filtering, text databases, digital libraries, and other aspects, but the problem of feature selection is as much, or more important than text categorization.

In this thesis, we will use Reuters-21578, 20-Newsgroups and The 4-Universities datasets to perform experiments. During the first step of thesis we will study some important topics ranging from collecting data, to organizing data and ultimately using the organized data, to conduct tests using the feature selection metrics efficiently.

The idea behind of feature selection is to use some measure to decide the significance of words that can keep informative words, and remove non-informative words that can then help the text classification system assign a category of a document.

The following feature selection metrics will be studied in this thesis; DF, MI, CHI, GSS Coefficient. After that, selected features will be used by classification process in Python and Scikit-Learn to get the best ML algorithms and the best performance of the system, by computing performance measures such as Accuracy, Recall, Precision and *F*-measure.

Finally, we will compare the usability of machine learning algorithms that are actively used for text classification and spot the best classifiers to use in machine learning for text categorization.

1.3 Outline

This thesis is divided into five chapters, Chapter 1 is an introduction to text categorization. It includes the literature review, motivation and outline of the thesis. Chapter 2 contains the brief definition of text categorization, document representation, term weighting and feature selection metrics. Chapter 3 describes the used tools, datasets, preprocessing techniques, classification algorithms, document representation, feature selection methods and performance evaluation metrics. Chapter 4 is the experimental results and discussion part. Chapter 5 presents the conclusion and future work.

Chapter 2

TEXT CATEGORIZATION

As already mentioned in the introduction part of this thesis, in line with the improvements in technology, text documents should be categorized in order to access the desired information in a more organized way. TC is the separation of texts into categories according to their characteristics [12].

The representation way of the documents to be classified is the most crucial factor affecting the performance of the classification process in TC. The natural consequence of this is that the selection of informative features improves the classification performance [6].

Text categorization is generally used to group materials such as news, articles, etc. according to their authors or genres, to filter spam emails and to improve indexing for search engines.

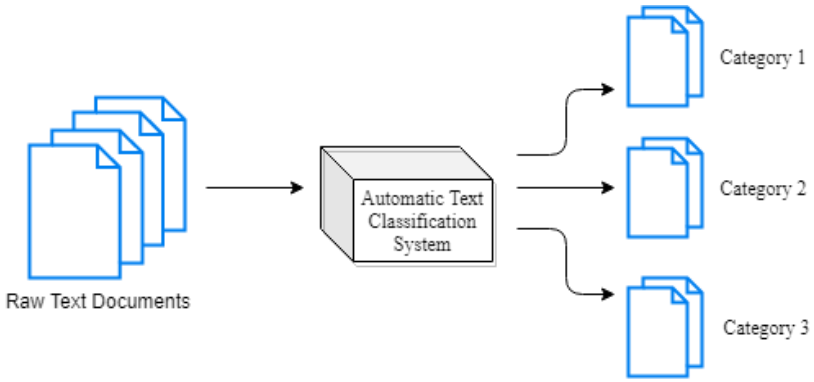

Figure 2.1: Illustration of TC

Text categorization plays a significant role in areas such as ML and generally includes three approaches: manual, conditional, and automatic classification. This thesis focuses on the automatic categorization approach [45].

Automatic categorization is to perform ML using training data. The training data set can be defined as the collection of good documents from each category. The aim is to estimate the class of a document by considering the information gathered from the training set when a document is submitted to the system. In the literature, ML method is seen in two different kinds as "supervised" and "unsupervised" [12].

2.1 Supervised Text Categorization

In the supervised ML approach, the category of material to be used as training data is predetermined. In this type of learning process, a model is derived from the training set that can estimate the category of a given material for the classification task. SVMs for supervised learning, DTs, kNN, NNs and random forest (RF) etc. algorithms are used.

2.2 Unsupervised Text Categorization

In this approach, unlike supervised learning, the categories of training data are not known. Unsupervised learning methods are intended to determine which objects are grouped as a class. Clustering algorithms such as kNN, *k*-means are used for grouping objects.

2.3 Text Preprocessing

As already mentioned in the introduction part of this thesis report, the first step of proper classification using state-of-art classification algorithms is to extract unnecessary data.

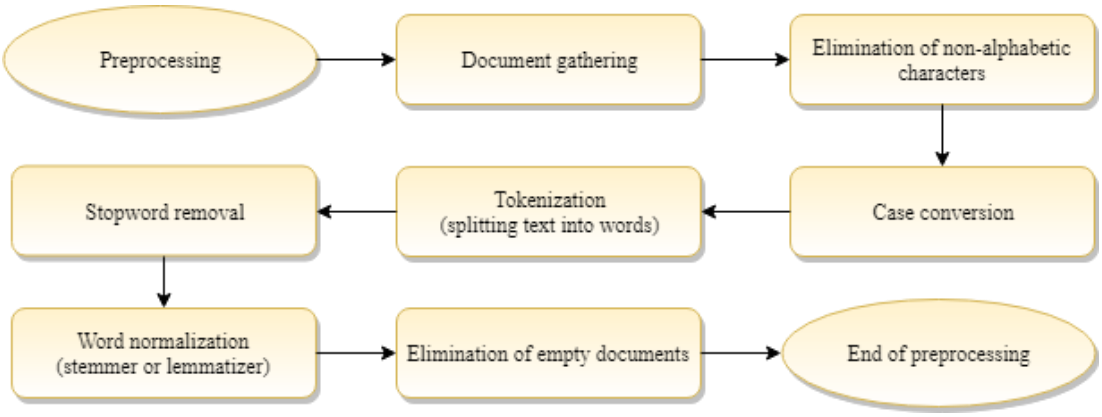

Figure 2.2: Steps of Text Preprocessing

Figure 2.2 illustrates that steps of text preprocessing. At the beginning of preprocessing after documents have been gathered, the non-alphabetic characters such as numbers, punctuations, and special characters, dates, and tags are removed. This process followed by the case conversion, which is necessary because computers do not examine words semantically and cannot distinguish between upper and lower case. Otherwise, the word "feature" and "Feature", which are actually the same, will be treated as two different tokens. After case normalization, the text is divided into words namely; tokenization is performed. This will provide convenience in the future to weight the features. Subsequently, words that frequently used in the structure of natural languages, such as conjunctions, markers which do not actually contribute to the content, removed from the text. In the structure of natural languages, words can exist in many different forms. For example, the words "computer", "computing" and "computation" actually originate from the same "comput" root. Since the preprocessing method actually aims to reduce the feature space to be used as input in the classification system, words are reduced to root forms by using stemmer or lemmatizer algorithms. The last step is to eliminate such documents whose contents become empty as a result of all these steps.

In this thesis, we used three different stopwords lists from three different sources such as Natural Language Toolkit (NLTK) [46], SMART [47] and a specific list to Reuters-21578 corpus [48] for eliminating common and frequent words and finally Porter's stemmer algorithm [49] is used to reduce words to their root forms.

2.4 Document Representation

As mentioned earlier in this thesis report, in fact, the TC process is to ask the computer to mimic people's behavior. Since computers cannot consider the texts in semantic point of view like humans and natural language texts are generally in unstructured form, it is necessary to present documents to computers in structured data form.

VSM is used commonly to represent documents in a structured way for computers. Here, preprocessed documents represented as a vector *D*, and each dimension of this vector corresponds to a term found in the document collection. After processing raw documents, we represented them in a form so that the computers can understand and can calculate it using term weighting methods. In Figure 2.3 below, the vector representation of documents is illustrated.

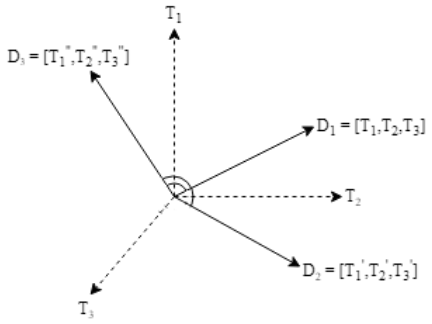

Figure 2.3: Document Representation as Vectors

2.4.1 Bag of Words Model (BoW)

In the thesis study, the BoW model is used in the VSM to express the documents as a vector consisting of words. In this model, each term represented as one word. Furthermore, according to this model, the vectors D_1 = ["t_1", "t_2", "t_3"] and D_2 = ["t_3", "t_1", "t_2"] are considered the same vectors. Normally, in natural languages, words come together in a different order and, sentences created in a different sense, but this is not the case in the BoW model. Regardless of the order of vectors, they are similar in terms of the same words they contain.

2.5 Term Weighting

As already mentioned in Document Representation section, computers cannot interpret the texts as people do, but instead, they can bring together similar materials by applying statistical processes. Thus, by using term weighting methods, words can be interpreted by computers for the classification process. After the term weighting performed, each document is represented as a vector *D*, as shown in equation (2.1).

$$D = [W_1, W_2, \dots, W_n] \tag{2.1}$$

Where, W_j is the weight of corresponding term *j* in the document *D*.

According to the study by Zobel J., Moffat A. [50], there are three general assumptions:

- The number of words is not a determinant of the importance of the document when documents containing more words are compared with documents containing fewer words.
- Words that are less common in a document are no less important than common words.

- Words that appear more than once are no less important than the word seen once.

Given all three assumptions above, it can be said that TF-IDF, which is a commonly used and successful method for term weighting purposes. It also balances the weight of the term by considering these three assumptions as follows.

Length normalization corresponds to the first item, TF to the second item and DF to the third item.

Taking these three assumptions into consideration, the weight of a term is calculated with TF-IDF formula from study of Zobel J., Moffat A. [50] as follows.

$$W_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i}) \tag{2.2}$$

Where;

$W_{i,j}$ is the weight of the term i in the document j.

$tf_{i,,j}$ is the number of occurrences of term i in the document j.

df_i is number of documents that contains term i.

N is the total number of documents in the corpus.

TF-IDF determines the importance of a word according to its frequency in the document, and also considers how often it occurs in the corpus. Although there are various weighting methods such as Boolean, DF, TF, we will focus on the use of TF-IDF weighting method in this thesis.

When the term weighting and representation methods are combined, the documents are ready for input to the classification algorithm. The dataset is seen as the document - term matrix in the figure below.

Figure 2.4: A Document Term Matrix

As shown in Figure 2.4, the rows indicate the documents and the columns demonstrate the weights of the terms corresponding to the documents.

2.6 Feature Selection

Each term in a text document is treated as a feature. A text corpus contains many features, and most of these features are non-informative and not discriminative in terms of category.

The primary purpose of the feature selection is to reduce the dimensionality of the feature space by selecting a subset of features that are discriminative and informative from the original feature set.

This subset selection process can be defined as selecting the best features with the highest score according to the result of a feature selection metric. A well-applied

feature selection can lead to improve the efficiency of categorization or will result in less information loss [5].

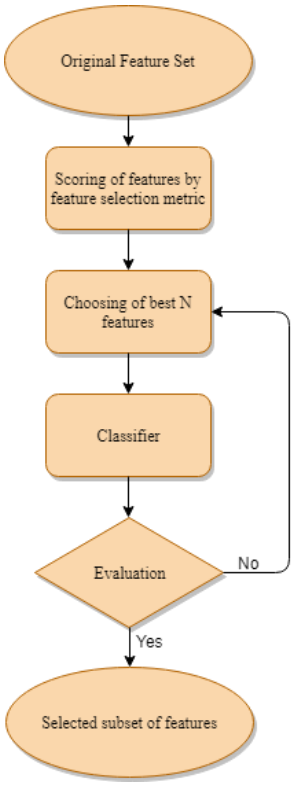

Figure 2.5: Flowchart of Feature Selection

As shown in Figure 2.5, each feature in the original feature set is scored and ranked by a feature selection metric. Among the ranked features, *N* features with the highest score are selected. Classification is performed with the selected subset and the result is evaluated by performance evaluation metrics such as accuracy score and *f*-measure. If the performance is satisfactory as a result of the evaluation, the subset of the selected feature is used to train the model. Otherwise, operations are repeated by selecting different subsets.

2.7 Feature Selection Metrics

In this section, the four feature selection methods that we used in the thesis study: DF, CHI, MI and GSS Coefficient and their formulas will be explained.

Table 2.1: Contingency Table for Feature Selection Metrics

	C_i	$\bar{C}_i$
t_k	*A*	*B*
$\bar{t}_k$	*C*	*D*

Table 2.1 above, shows the fundamental information elements for formulating feature selection metrics. Where:

A is the number of documents that belong to category C_i and contains term t_k.

B is the number of documents that do not belong to category C_i and contains term t_k.

C is the number of documents that do not contain the term t_k and belong the category C_i.

D is the number of documents that do not contain the term t_k and not belong the category C_i.

2.7.1 Document Frequency (DF)

DF is the number of documents in which the term occurs in a corpus. It is a simple approach to the selection of features; the idea behind is that, rare words are not discriminative for determining the category [51].

The document frequency of term t_i can be calculated with the equation (2.4) according to Table 2.1.

$$DF_i = A \tag{2.4}$$

2.7.2 Chi – Square (X^2 - CHI) Test

CHI can be considered as a compatibility test. It shows how much a term fits into a given probability distribution by the square of differences method. In terms of the feature selection method, this approach is based on measuring the dependence between

term t_k and category c_i. It has a zero value naturally when the t_k and c_i are independent [6].

The CHI score for term t_k, and category c_i can be calculated as in the equation (2.4) following formula by considering the information from Table 2.1.

$$X^2(t_k, c_i) = \frac{N(AD - CB)^2}{(A + C)(B + D)(A + B)(C + D)} \quad (2.4)$$

Where N is the number of documents in the training set.

2.7.3 Mutual Information (MI)

MI is a metric used for modelling the word associations in statistics. MI measures how much information is available between the terms and the category to determine the terms that can accurately represent a category and how the terms contribute to that category. It has zero value when the t_k and c_i are independent [5].

The MI score between t_k and c_i can be calculated by the equation (2.5) according to Table 2.1.

$$MI(t_k, c_i) \approx \log \frac{A \times N}{(A + C)(A + B)} \quad (2.5)$$

Where N is the number of training documents in the document collection.

2.7.4 Galavotti Sebastiani Simi (GSS) Coefficient

Galavotti Sebastiani Simi Coefficient is a simplified CHI statistic by completely removing of $\sqrt{N}$ factor and the denominator. Uchyigit G., and Ma M. Y. [9] removed the $\sqrt{N}$ factor because it is unnecessary and also denominator removed since it gives the high correlation score to rare words and categories [10]. GSS score of term t_k and category c_i can be calculated with the equation (2.6) by Table 2.1.

$$GSS(t_k, c_i) = (AD - CB) \quad (2.6)$$

Chapter 3

METHODOLOGY

This section provides information about the tools, datasets, methodology and performance measurement metrics that we used in our study.

3.1 Used Tools

3.1.1 Scikit-Learn

Scikit-Learn is an open-source ML library for Python programming language. It provides many tools, including the modules for preprocessing, classification, model selection [52].

3.1.2 Natural Language Toolkit (NLTK)

NLTK is a platform for Python to work with natural language. It provides corpus readers and text processing libraries including parsing, tokenizing, stemming, lemmatizing and part-of-speech tagging (POSTAG) [53].

3.1.3 SciPy

SciPy is an open-source module suite for Python. It provides powerful packages such as Pandas, Numpy, and Matplotlib for scientific purposes [54].

3.2 Datasets

3.2.1 Reuters-21578 ApteMod Version

The standard Reuters-21578 dataset consists of a total of 21578 documents, including documents without topic codes and typographical errors. For this reason, a new subset called "ApteMod" is created by removing such documents from the standard dataset [14].

The Reuters-21578 ApteMod is built for benchmarking of TC tasks. It is a collection that consists of 7769 training and 3019 test documents collected from the Reuters Financial Newswire Service. There are 90 categories in that corpus, and a document can belong to one or more categories. Since it is commonly used in the studies which are in the literature, we decided to use this dataset to compare our results with other studies.

We randomly choose ten topics from the dataset. Since the dataset ordinarily suitable for the classification of the multilabel classification tasks, a document can have one or more labels. For this reason, if a document has more than one label, the documents are multiplexed and rearranged into one label each. For example, if a document *D* has "acq", "earn" and "coffee" tags, this document is placed three times to dataset with different tags *D* - "acq", *D* - "earn", *D* - "coffee". Thus, we converted the dataset to a single label form.

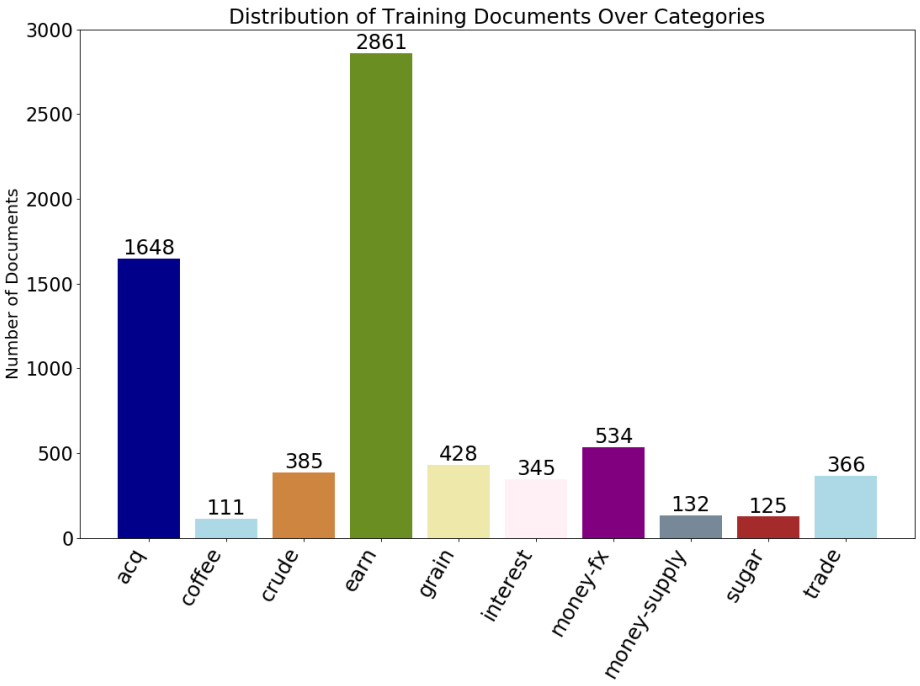

Figure 3.1: Distribution of Training Documents in Reuters Dataset

Figure 3.1 shows the distribution of training documents over the category names after the preprocessing process, including the elimination of empty documents.

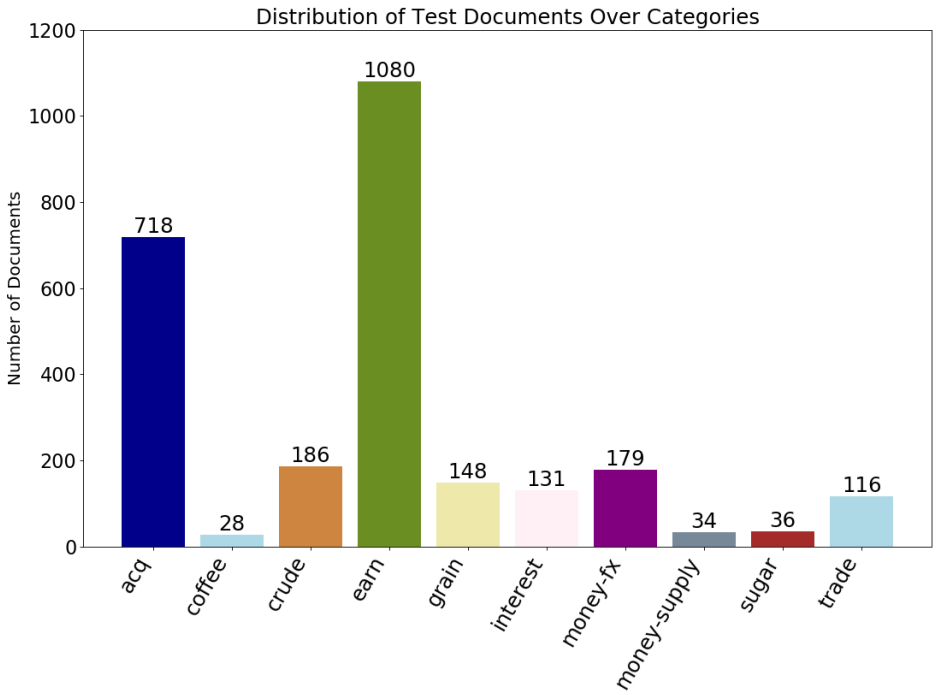

Figure 3.2: Distribution of Test Documents in Reuters Dataset

Figure 3.2 illustrates the distribution information of text documents over selected ten categories.

At the end of the dataset pre-processing, we reformed the Reuters dataset as; 6935 training documents, 2656 test documents, and 16021 features.

3.2.2 20 Newsgroups Dataset

The 20 Newsgroups dataset consists of 18846 documents in total. The documents are organized into 20 different groups. Since the dataset has already train/test split parts (60% and 40% respectively) by their dates, we used “bydate” version [55] in our thesis study [16].

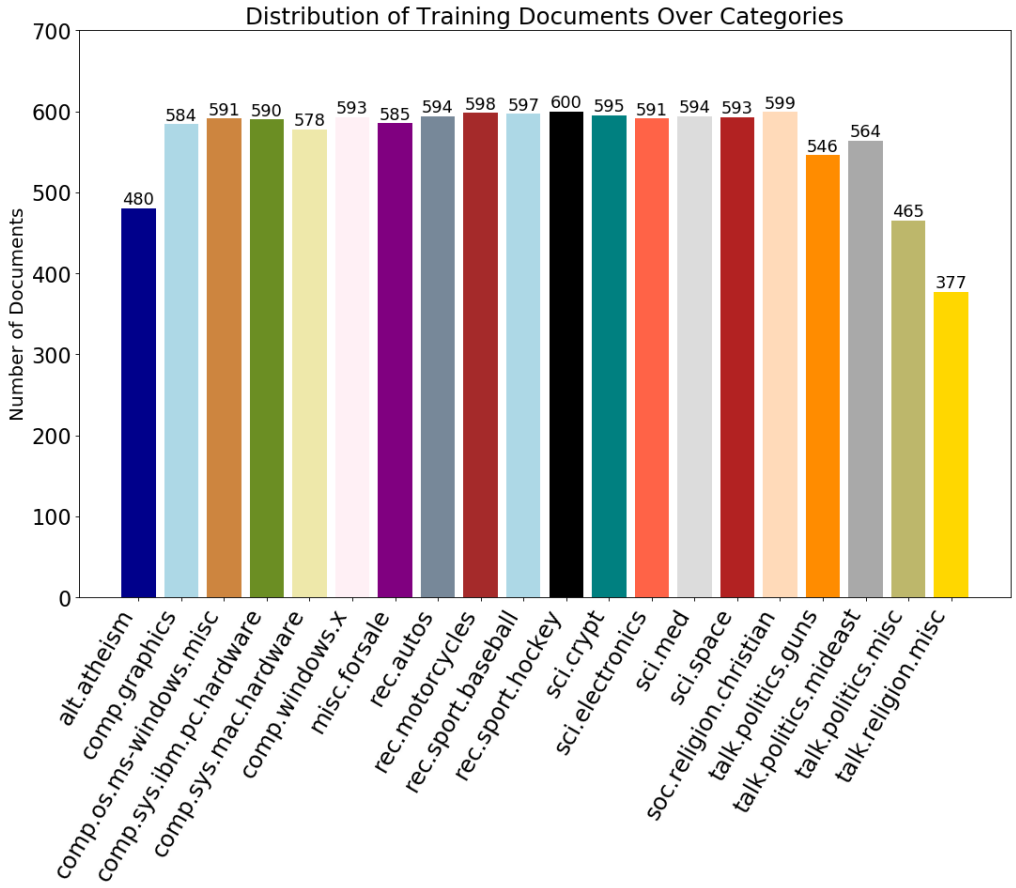

Figure 3.3: Distribution of Training Documents in 20 Newsgroups Dataset

The Figure 3.3 depicts the distribution of training documents according to categories in the original dataset.

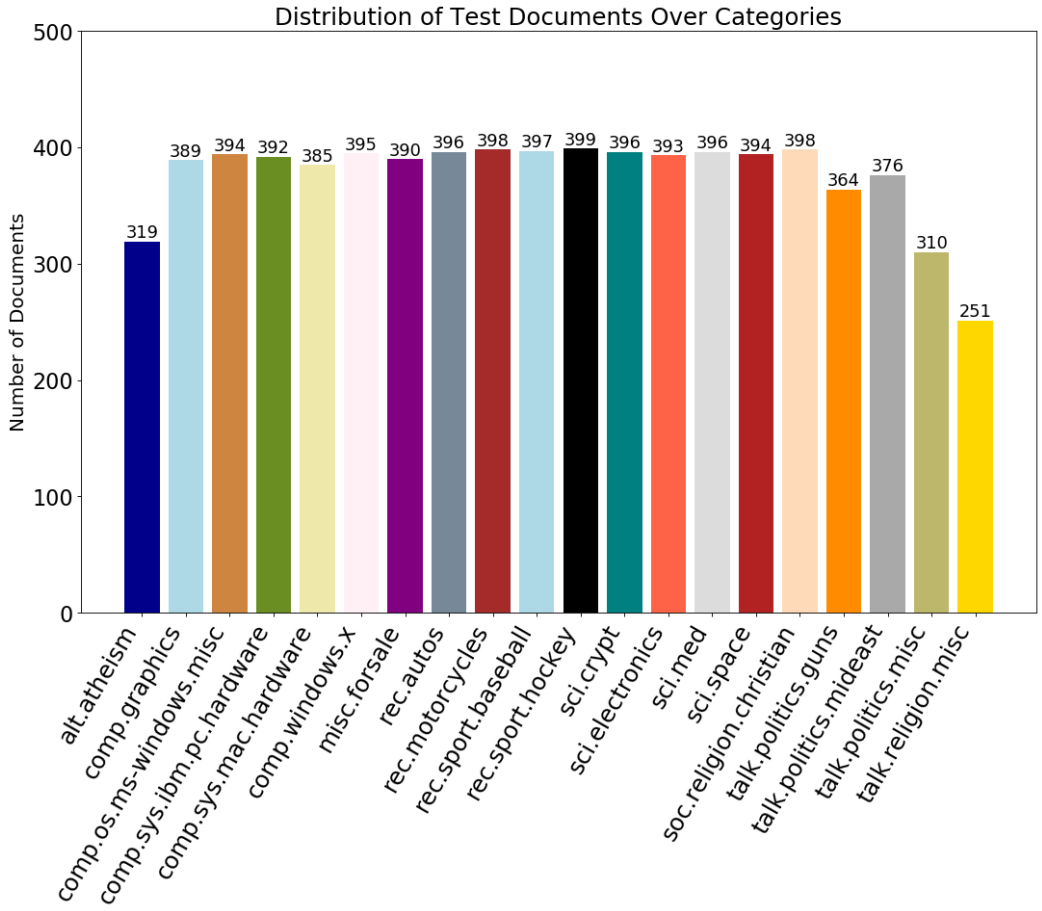

Figure 3.4: Distribution of Test Documents in 20 Newsgroups Dataset

Figure 3.4 shows the number of documents in the test set according to the categories.

Due to the computation and memory limits of the computer, we chose ten most commonly used categories in order to facilitate the categorization procedure.

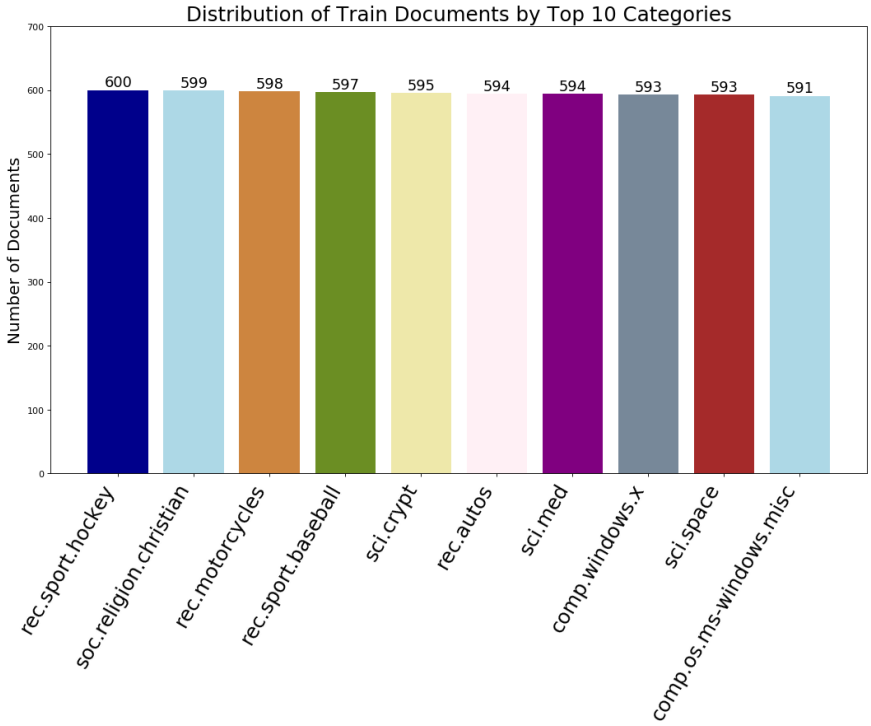

Figure 3.5: Document Distribution for Training Set by Top Ten Categories

Figure 3.5 illustrates distribution information of training documents in the selected categories.

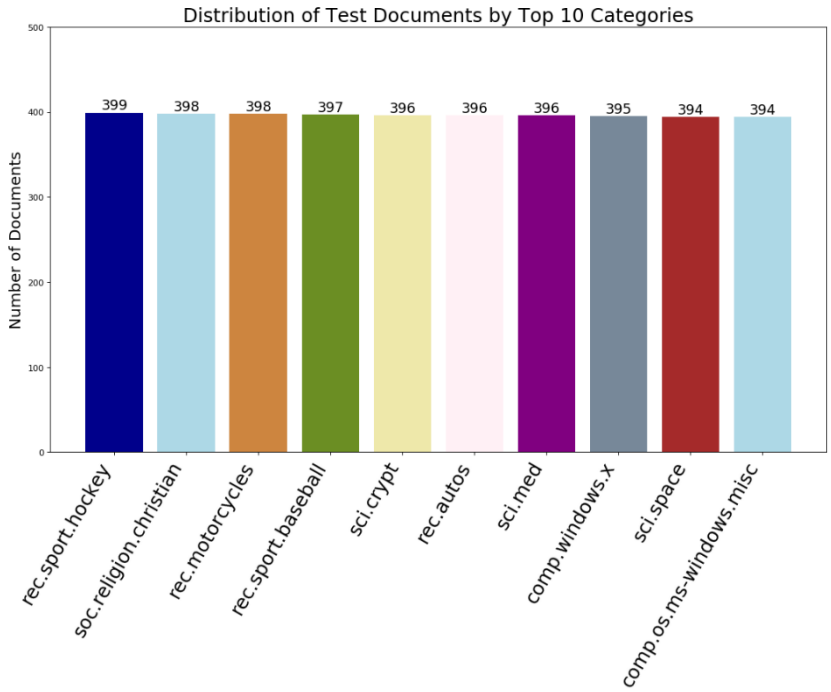

Figure 3.6: Distribution of Test Documents by Top 10 Categories

Figure 3.6 shows the distribution of documents in the test set over top ten categories. As can be seen from both Figure 3.5 and Figure 3.6, the training and test subsets are balanced in term of the number of documents. After the preprocessing steps, 9917 documents and 46232 features remained in the subset.

3.2.3 The 4 Universities Dataset

The 4 Universities dataset is a collection of web pages from the department of computer sciences of many universities by the CMU Text Learning Group [15].

The documents are grouped under seven different classes in the data set, but the "staff" and "department" classes contain only a few documents, so they are excluded from the data set. Documents belonging to the class "Other" are also very different from each other, and we did not consider them into account. After removing these classes, the distribution of documents with class names are depicted in Figure 3.7 below.

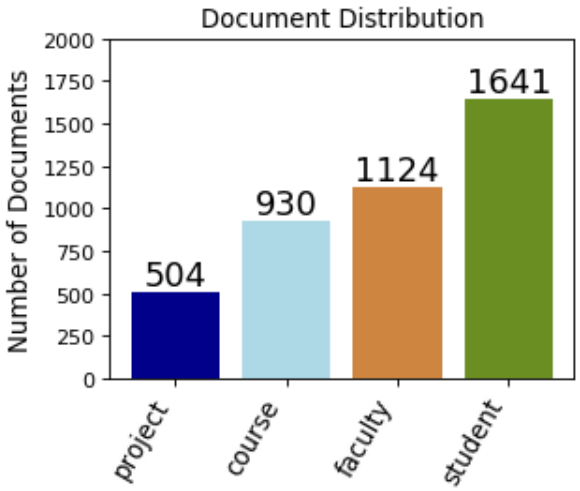

Figure 3.7: Document Distribution in 4 Universities Dataset

Since there is no train/test split, we split the dataset into training and test sets. We randomly chose 67% of documents for the training set, and we used the remaining as a test set.

Table 3.1: Number of Documents after Splitting

Class Name	Training Documents	Test Documents	Total Documents
project	329	175	504
course	610	320	930
faculty	752	372	1124
student	1122	519	1641
Total	2813	1386	4199

Table 3.1 gives detailed information on the number of documents for each class. After the train/test splitting process, the dataset contains 4199 documents and 48633 features.

3.3 Preprocessing

In text categorization, applying of preprocessing to texts reduces the computational load for predicting the classes of texts. In the preprocessing stage, we removed the conjunctions, prepositions, punctuations, and words that are not discriminative for categorization from the texts.

We conduct a series of tests to find out the effects of different stopword lists and term elimination process.

Table 3.2: Effects of Stopwords Lists

Stopword list	Classification Accuracy (SVM Linear)	Classification Accuracy (MNB)
None	0.9427	0.8234
NLTK	0.9175	0.8532
SMART	0.9401	0.8539
Reuters-21578 Corpus Specific	0.9179	0.8238
Combination of NLTK, SMART and Reuters-21578	**0.9430**	**0.8550**

Table 3.2 shows the effects of three different stopword lists. According to the test results, we decided to use a combination of three different stopwords lists: SMART, NLTK, and Reuters-21578 corpus specific. As mentioned in the study of Leopold, E., & Kindermann, J. [28], there is no need to remove stopwords or rare words when the SVM is applied to text classification.

During the preprocessing, we divided the texts into words/tokens, and then we performed case normalization. Since computers handle the words statistically, this normalization needs to be applied. As a second step, we eliminated the punctuation marks, stopwords, and abbreviations of month names. In the third step, non-alphabetic characters removed from the words by using regular expressions.

In the preprocessing stage, hyphenated words such as "tie-up" and abbreviations such as "U.S." are considered as a single term combined with underscores. In the last step of text processing, words are reduced to their root forms by using Porter Stemmer algorithm and documents that become empty after applying the preprocessing also eliminated from training and test sets. Thus, words those are in different forms derived from the same root, reduced to one word, and we observed a significant decrease in the general vocabulary.

3.4 Document Representation and Term Weighting

In this thesis, a vector corresponding to each document is generated, and the documents are represented with these vectors. The vectors are generated using a generic vector (vocabulary vector) that represents all of these words singularly using only the words found in the training documents. In other words, the length of all document vectors is equal to the length of the vocabulary of the training set documents.

The following example shows that how we generated the document vectors:

Document 1: “laboratori experiment softwar perform experiment laboratori”

Document 2: “research decemb experiment softwar”

Assume a dataset that consist of two documents as above, the vocabulary vector for constructing the document term matrix will contain following six terms:

vocabulary vector: ["decemb", "experiment", "laboratori", "perform", "research", "softwar"].

Once the vocabulary vector created, the term frequency of the words in that document is computed simply for each document. Since the term frequency has local scope, it can be easily calculated by counting the number of times a word appears in a document [50].

Table 3.3: An Example to Term Frequencies

	decemb	**experiment**	**laboratori**	**perform**	**research**	**softwar**
Document 1	0	2	2	1	0	1
Document 2	1	1	0	0	1	1

Table 3.3 shows that a document – term matrix with frequencies of terms. As seen in the table, the term “experiment” appears two times in *Document 1* and it appears once in *Document 2*.

The document frequencies of terms can be computed by counting the number of different documents the term appears in [50].

Table 3.4: An Example to Document Frequency (DF)

Term	Document Frequency
decemb	1
experiment	2
laboratori	1
perform	1
research	1
softwar	2

As seen from the Table 3.4, the terms “decemb”, “laboratori”, “perform”, “research” appeared in only one document and the terms “experiment” and “softwar” appeared in two different documents.

$$idf_i = \log_e\left(\frac{(1+N)}{(1+df_i)}\right) + 1 \tag{3.1}$$

The “TfidfVectorizer” in the “Scikit-Learn” library computes the IDF of the term t_i by using equation (3.1) above [52].

Table 3.5: An Example to IDF Values

Term	Inverse Document Frequency (IDF)
decemb	1.4054651081081644
experiment	1.0
laboratori	1.4054651081081644
perform	1.4054651081081644
research	1.4054651081081644
softwar	1.0

Table 3.5 shows that calculated IDF values of each term in the vocabulary vector. The TF-IDF calculation can be done by multiplying TF values with IDF values.

Table 3.6: An Example of TF-IDF Calculation

	decemb	experiment	laboratori	perform	research	softwar
Document 1	0	2	2.81093	1.40547	0	1
Document 2	1.40547	1	0	0	1.40547	1

Table 3.6 provides information on the calculated TF-IDF weights according to the example dataset.

The datasets can contain many documents with different sizes. In such cases, documents with more words are more likely to have frequent terms, and since such terms also have an effect on weight calculation, documents with more words can dominate the prediction of classes. In order to avoid this situation and to consider all documents fairly, the length of the documents should be normalized [56].

In the thesis study, we did tests by using TF-IDF vectorizer module from Scikit-Learn library with combination of different normalization and weighting parameters to investigate the effects of normalization methods.

Description of the parameters of TF-IDF vectorizer as following;

- **sublinear_tf:** applies the sublinear TF scaling as in equation (3.2).
- **smooth_idf:** since IDF is zero when there is a document containing all the terms in the collection, it prevents from the zero division by adding one to DF as in equation (3.1) when the parameter value is “True” otherwise IDF weight can be found by equation (3.2).
- **use_idf:** enable or disable the IDF weighting.
- **norm:** it denotes that the normalization. Parameter values can be “l1”, “l2” or “None”. When the value is “None”, it disables the normalization factor.
- **vocabulary:** is the parameter to use the specific term set.

The *sublinear_tf* can be computed with equation (3.2) [50].

$$sublinear\_tf = 1 + \log(tf) \tag{3.2}$$

The IDF weight of a term *t* can be computed by equation (3.3) below [56].

$$idf(t) = \log\left(\frac{n}{df_t}\right) \tag{3.3}$$

Where, *n* is the total number of documents in the corpus.

The l_1 normalization can be applied according to following equation (3.4) [56].

$$l_1\ norm = \frac{D}{\sqrt{\sum_{i=1}^{m} t_i}} \tag{3.4}$$

The l_2 normalization can be applied by using the equation (3.5) below [56].

$$l_2\ norm = \frac{D}{\sqrt{\sum_{i=1}^{m} t_i^2}} \tag{3.5}$$

Where;

D denotes that the document vector.

t_i denotes that the i_{th} term in the document vector D.

m denotes that the length of document vector.

Table 3.7: Comparison of Normalization Methods with Term Weighting

TfidfVectorizer				SVM Linear Kernel	MNB
Norm	**smooth_idf**	**sublinear_tf**	**use_idf**	**Classification Accuracy**	
L1	True	True	True	0.7361	0.4202
L1	True	True	False	0.7692	0.4401
L1	True	False	True	0.7583	0.4232
L1	True	False	False	0.8208	0.4443
L1	False	True	True	0.7349	0.4198
L1	False	True	False	0.7692	0.4401
L1	False	False	True	0.7564	0.4221
L1	False	False	False	0.8208	0.4443
L2	True	True	True	0.9386	0.8509
L2	True	True	False	0.9401	0.8276
L2	**True**	**False**	**True**	**0.9416**	**0.8554**
L2	True	False	False	0.9412	0.8340
L2	False	True	True	0.9386	0.8505
L2	False	True	False	0.9401	0.8276
L2	False	False	True	0.9387	**0.8554**
L2	False	False	False	0.9412	0.8340
None	True	True	True	0.9334	0.8410
None	True	True	False	0.9367	0.8513
None	True	False	True	0.9300	0.8209
None	True	False	False	0.9285	0.8470
None	False	True	True	0.9337	0.8491
None	False	True	False	0.9367	0.8411
None	False	False	True	0.9296	0.8383
None	False	False	False	0.9285	0.8470

According to the results given in Table 3.7, l_2 normalization with TF-IDF weighting give the best result. We decided to use l_2 normalization and TF-IDF weighting in our experiments. The table provides information on different term weighting methods and normalization techniques. As seen from the table the usage of TF weighting alone gives good performance as stated in previous researches [22] [30] [12] and we verified it. Since the “smooth_idf” parameter of “TfidfVectorizer” considered only when the “use_idf” parameter is “True”, the results are same for [“True”,”False”,”False”] and [“False”,”False”,”False”] parameter settings.

After obtaining the document vector *D* with the TF-IDF weights, we applied the l_2 normalization to each document vector by using equation (3.5). The sample document – term matrix can be shown in Table 3.8 below.

Table 3.8:Normalized Document-Term Matrix using TF-IDF Weights

	decemb	**experiment**	**laboratori**	**perform**	**research**	**softwar**
Document 1	0	0.518534	0.728781	0.364391	0	0.25927
Document 2	0.576152	0.409937	0	0	0.576152	0.40994

We also performed experiments with DF weighting method. DF weights of the terms can be calculated as in Table 3.4 above. We built a global dictionary which consists of all the features to use the DF weighting method. Then, DF values placed into the dictionary for the corresponding term. Finally, the document term matrix is constructed by replacing the corresponding terms (if exist) with DF values into the dictionary as we did for TF-IDF.

3.4.1 Probability Based Term Weighting

Liu, Y., Loh, H. T., & Sun, A. [34], proposed a probability-based term weighting approach for classification of imbalanced texts. Since the Reuters-21578 dataset is imbalanced, we decided to test the proposed term weighting approach.

This approach is derived from TF-IDF weighting, but the IDF part changed with a weight for the strength of representing a specific category. Thus, the term weights can be given by category membership.

To apply the probability-based term weighting, firstly we calculated the *A*, *B*, and *C* values according to Table 2.1, then we used the given equation (3.6) and (3.7) in the [34] to calculate the term weights.

$$ntf_{i,j} = \frac{tf(t_i, d_j)}{max[tf(d_j)]} \tag{3.6}$$

$$W_{i,j} = ntf_{i,j} \times \log\left(1 + \frac{A}{B} \times \frac{A}{C}\right) \tag{3.7}$$

Where:

$ntf_{i,j}$ denotes that normalized term frequency of term t_i in document d_j.

$tf(t_i, d_j)$ denotes that the term frequency of term t_i in document d_j.

$max[tf(d_j)]$ denotes that the maximum term frequency in document d_j.

$W_{i,j}$ is the normalized term frequency of term t_i in document d_j.

Local weight: is the TF of term t_k. In other words, it is the weight that is based on appearance of the t_k in the document *D*.

Global weight: is the IDF value of the term t_k. It defines the contribution of term t_k to a specific document in a global sense.

The two ratios are used in the term weighting approach:

A/*B*: term t_k is good to predict the category for document d_j. It tends to be higher if the term t_k is relevant to category c_i.

A/C: when two terms t_k, t_l and a category c_i taken into consideration, the term with a higher value of *A*/*C* ratio is more likely to represent category c_i.

We tested the probability-based weighting method on three different datasets. Although it gave good results on Reuters-21578, the results are bad for the other two datasets. We have confirmed that this term weighting scheme works well only for imbalanced datasets.

3.5 Classification Algorithms

3.5.1 Support Vector Machines (SVMs)

Support Vector Machines are used to classify linear and nonlinear data. The main idea in the SVM method is to have a plane that optimally separates the positive and negative samples. Through its high generalization capabilities, the plane to separate the samples can also be easily found on high dimensional data. Since text data has high a dimensional feature space, it outpaced many learning methods in TC [12].

As seen in Figure 3.8 below, in order to find the optimal plane (hyperplane) to separate the data, parallel lines that define the boundary of the data points are determined. These lines are referred to as support vectors. The distance between the support vectors is maximized, and the hyperplane is placed in the middle of this maximized distance [12]. Here, the goal is to select the hyperplane to maximize the margin from both training data classes. Maximizing the distance between each class ' closest points and the hyperplane would lead in an ideal hyperplane separation.

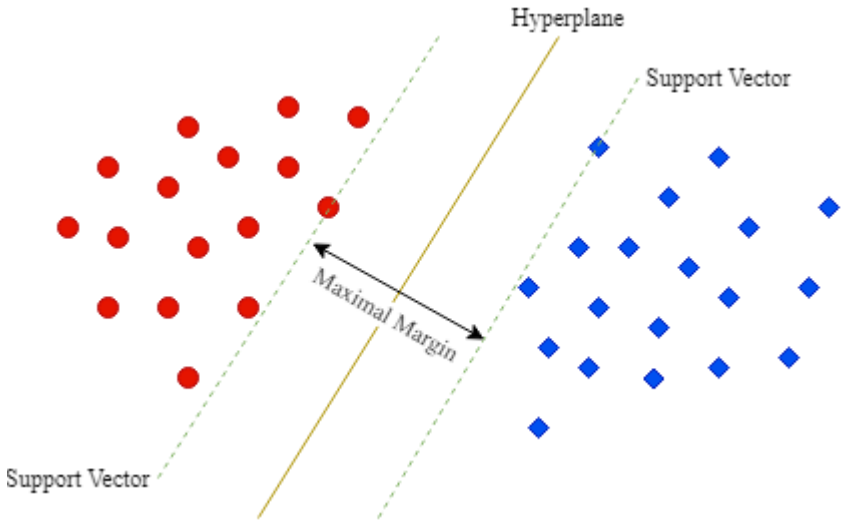

Figure 3.8: Finding an Optimal Hyperplane in SVMs

While a line between two data classes can separate linear data, nonlinear data can be separated from each other by expressing in high dimensional plane.

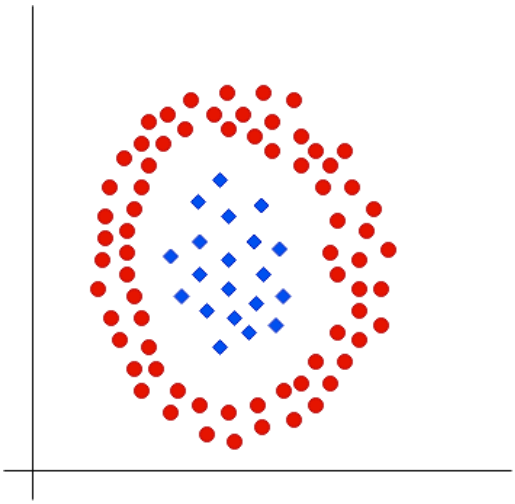

Figure 3.9: Non-Linearly Separable Data Points

Figure 3.9 illustrates the non-linearly separable data. To separate classes, another kernel can be used such as Radial Basis Function (RBF).

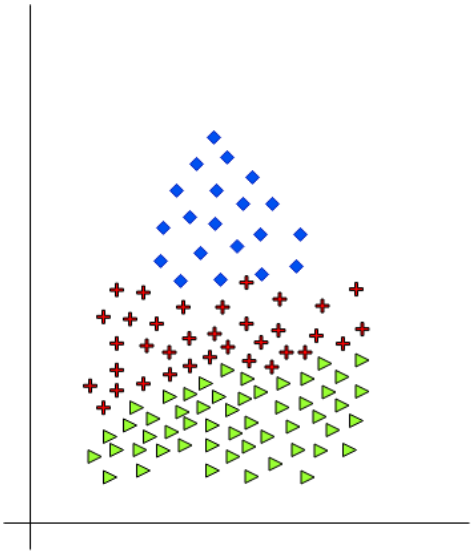

Figure 3.10: Non-Linearly Separable Data in High Dimensional Plane

Figure 3.10 depicts the non-linearly separable data in a high dimensional plane. The data points can be separated by RBF kernel.

Figure 3.11 below illustrates the one-vs-rest approach for SVM. This approach is used in our experiments as a decision function parameter for SVM. OVR approach trains one classifier per class.

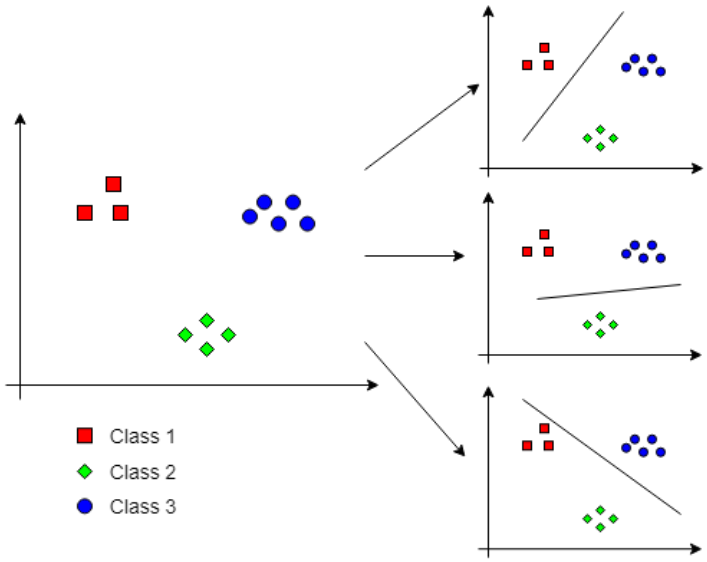

Figure 3.11: Illustration of One-vs-rest Approach

3.5.2 Naive Bayes (NB)

NB is a statistical classification algorithm that uses Bayes theorem. NB classifier is efficient and sensitive in feature selection, although it is essential for feature selection, it does not give good results in high dimensional data [11].

NB requires many calculations because of the possibility of data on categories. The probability that a document belongs to a category can be calculated with equation (3.8) [57].

$$argmax\ P(c_i)\prod_{k=1}^{n} P(t_k|c_i) \tag{3.8}$$

Where;

n: denotes the total number of terms in the class c_i.

$P(c_i)$: denotes the probability that a document belongs to class c_i. $P(c_i)$ can be found by proportioning the number of documents belonging to class C to the total number of documents in the dataset.

$P(x_k|c_i)$: indicates the probability that the term t_k is in the class c_i and is found by the ratio of the number of term t_k appears in the class c_i to the total number of words.

3.5.3 Multinomial Naive Bayes (MNB)

MNB is a probabilistic method that considers the frequency of each term in the documents. It is considered more appropriate for TC [58].

MNB assigns the test document d_i to the class c with the highest probability by using equation (3.9) where C is the set of categories, and N is the size of vocabulary [59].

$$P(c|d_i) = \frac{P(c)P(d_i|c)}{P(d_i)}, c \in C \tag{3.9}$$

The prior probability $P(c)$ can be found by dividing the number of documents belonging to class c by the total number of documents in the dataset. $P(d_i|c)$ is the probability of obtaining a document d_i in class c, and it can be calculated by equation (3.10) below.

$$P(d_i|c) = \left(\sum_n f_{ni}\right)! \prod_n \frac{P(t_n|c)^{f_{ni}}}{f_{ni}!} \tag{3.10}$$

f_{ni} is the number of term t_n in document d_i and $P(t_n|c)$ the probability of term t_n in class c. The posterior probability can be found from the training document by equation (3.10).

$$\hat{P}(t_n|c) = \frac{1 + F_{nc}}{N + \sum_{x=1}^{n} F_{xc}} \tag{3.10}$$

F_{xc} is the frequency of term *x* in the training set of documents belonging to class *c* and in the case where the frequency of a term is zero, 1 is added to the number of each term to avoid the zero-division problem. Normalization factor $P(d_i)$ which is in the equation (3.9) can be calculated by equation (3.11).

$$P(d_i) = \sum_{k=1}^{|C|} P(k)P(d_i|k) \tag{3.11}$$

3.5.4 Gaussian Naive Bayes (GNB)

GNB is a probabilistic approach that uses the Gaussian probability distribution. It is based on the calculation of prior and posterior probability according to the training set and test set [60]. The prior probability of a class can be calculated using equation (3.12) below.

$$P(c) = \frac{Number\ of\ instances\ in\ class\ c}{Total\ number\ of\ instances\ in\ the\ dataset} \tag{3.12}$$

The posterior probability can be calculated an in equation (3.13).

$$\hat{P}(x|c) = \prod_{k=1}^{n} P(x_k|c) \tag{3.13}$$

Where:

x is the test data, x_k is the features in the test data x and $P(x_k|c)$ is the conditional probability of the test data in the class c.

The conditional probability of the test data can be calculated as in equation (3.14)

$$P(x_i|c) = \frac{1}{\sqrt{2 \times \pi \times \sigma_{x_i,c}^2}} \times exp\left(-\frac{\left(x_i - \bar{x}_{i,c}\right)^2}{2 \times \sigma_{x_i,c}^2}\right) \tag{3.14}$$

Where:

x_i denotes the i^{th} feature of the test data, c indicates the class and $\sigma^2$is the variance.

The conditional class probability of the test data can be calculated with Bayes Theorem (3.15).

$$P(c_i|x) = \frac{P(x|c_i) \times P(c_i)}{\sum_{j=1}^{n} P(x|c_j) \times P(c_j)} \quad (3.15)$$

Where x is the test data, c_i is the i^{th} class in the class set and n is number of classes.

After the calculations for each test instance and class c, the class with highest probability will be assigned to the test data.

3.5.5 K Nearest Neighborhood (kNN)

The kNN classification method is widely used in TC because of simplicity [61]. It does not create a set of rules or functions, such as other classification algorithms, before estimating the class of a test document. Therefore, it is more efficient in terms of training time, but the classification process takes a long time as it requires recalculation for each sample during the prediction.

In this method, categorization is performed by finding the nearest samples to the sample to be categorized with a distance measurement [13]. The label of the dominant class is assigned from the closest points to the given instance. When the number K is equal to 1, the class of the most similar sample to the given instance is assigned [57].

The distance between two data points $D_1 = [t_1, \dots, t_n]$ and $D_2 = [t'_1, \dots, t'_n]$ can be calculated by Euclidean distance as in equation (3.16) [57].

$$distance(D_1, D_2) = \sqrt{\sum_{i=1}^{n} (D_{1i} - D_{2i})^2} \quad (3.16)$$

The *k* number is generally chosen as an odd to find the dominant label between samples. In order to reduce the classification error, it should be selected as large and as well as smaller than the number of documents in the training set [57].

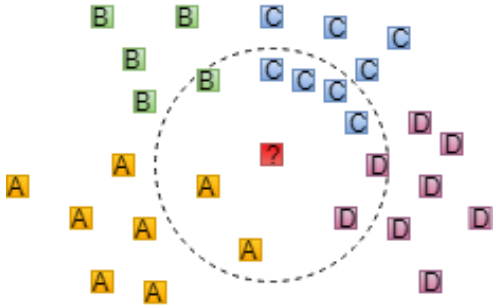

Figure 3.12: Classification with kNN

Figure 3.11 shows an example of classification with kNN. When the number *k* is selected as 9, the label of the dot shown in red will be assigned as "*C*".

3.5.6 Decision Trees (DTs)

In the DTs algorithm, the nodes represent the testing of features, the branches represent the result of the test, and the leaves represent the categories. DTs start with a root node that is the best predictor according to some measure. The data in the training set is divided according to the attributes in the root node [62] [63].

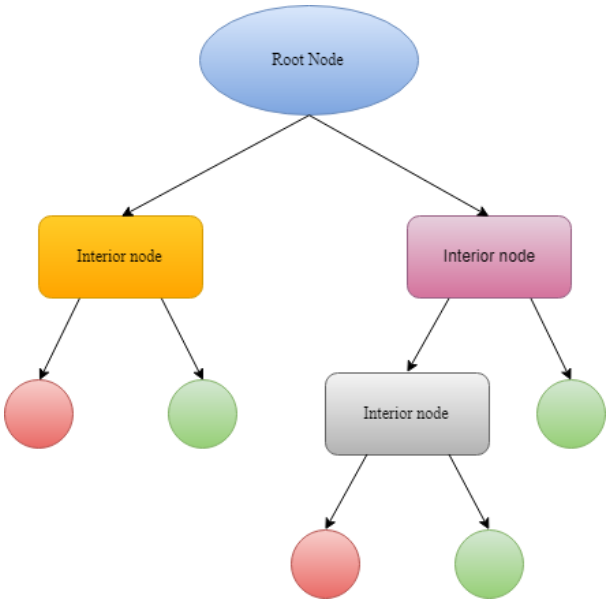

Figure 3.13: Illustration of Decision Tree

Figure 3.12 shows an example to a DTs. A root node contains the best estimator for predicting the target class chosen by some measure, and each arrow between the nodes represents test criteria. Interior nodes show the split attributes and leaf nodes are the predictions.

3.5.7 Random Forest (RF)

RF is an algorithm that can be used for classification and regression purposes. In our study, we focused on the usage of this algorithm for classification purposes.

The RF can be briefly described as a method consisting of a large number of DTs. For a classification problem, a large number of random subsets are selected from both the data set and the feature set. Then the training process is performed with creating different DTs [64]. Each DT makes an individual prediction and is used to assign the class that receives the most votes from the estimates made. Since the RF method works by selecting random subsets, it also provides a solution to the overfitting problem, which is the disadvantage of DTs.

Figure 3.13 illustrates the classification of an instance with RF classifier. An instance is given to a system of trained DTs. Each DT predicts the class of the given sample individually. Then the predictions evaluated by majority voting. The class that receives the most votes is assigned to the instance.

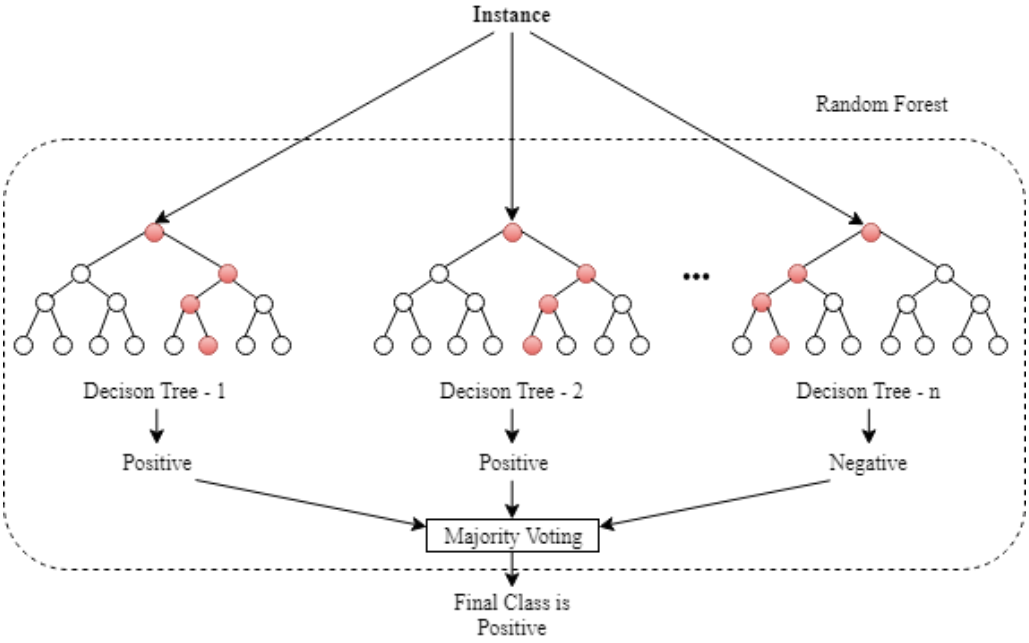

Figure 3.14: Random Forest

3.5.8 AdaBoost

AdaBoost aims to combine different weak learners / classification algorithms together for improving the performance of classification.

It trains each weak learner by using a set of training instances and gives a weight by adjusting them in each iteration. That means the AdaBoost algorithm trains weak learners in an iterative way, and the weights show the robustness of the learners. Weak learners can be classification algorithms such as DTs, NNs, etc. It combines the individual learners to construct a final predictive model [65].

The figures 3.14, 3.15, 3.16, and 3.17 below, give an illustrative example for constructing the final predictive model in AdaBoost algorithm.

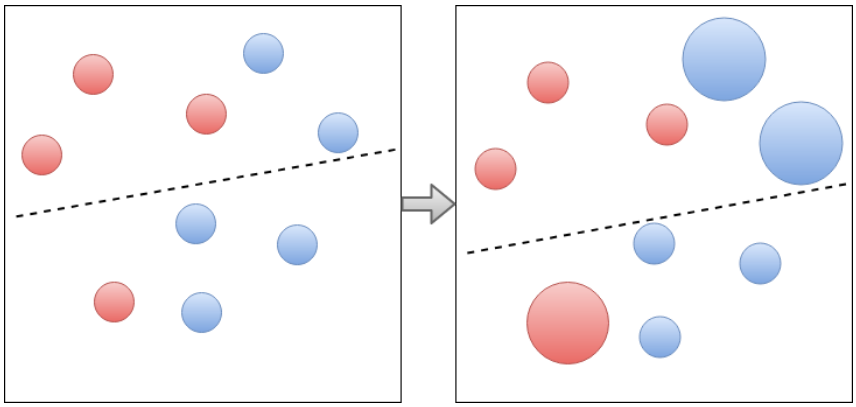

Figure 3.15: Weak Learner 1

Figure 3.14 shows the first weak learner in the AdaBoost algorithm. The first model, which is on the left side of the figure, makes classification and then the weights of misclassified points are increased as seen from the right side of the figure.

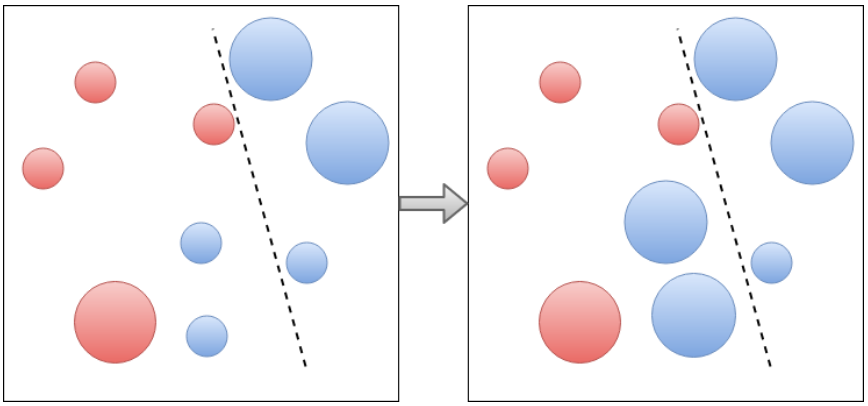

Figure 3.16: Weak Learner 2

As seen from Figure 3.15, the second weak learner performs classification, and again weights are increased for wrongly classified instances.

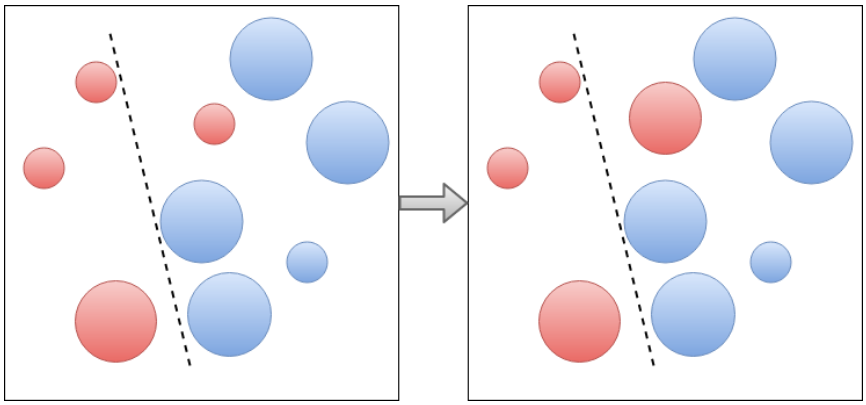

Figure 3.17: Weak Learner 3

The third weak learner increase the weights of misclassified samples as seen from the Figure 3.16.

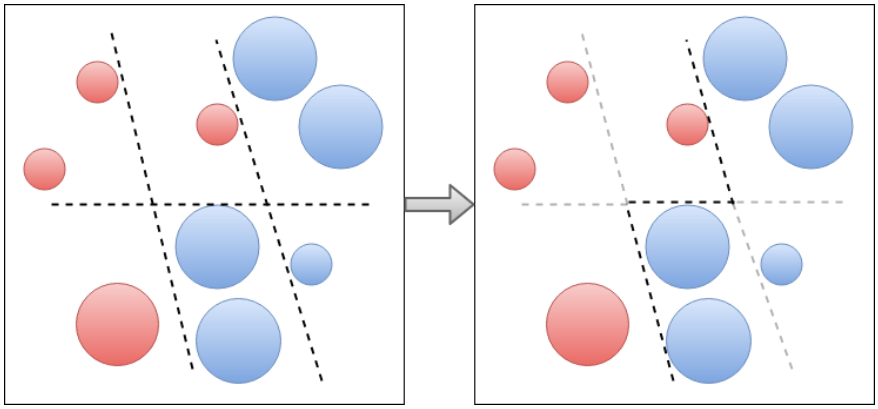

Figure 3.18: Combination of Weak Learners

Figure 3.17 above shows that the combination of weak learners to construct the final prediction model (on the right side of figure) in the AdaBoost algorithm.

3.5.9 Multilayer Perceptron (MLP)

MLP can be defined as a system of interconnected nodes/neurons. The nodes are connected with weights to each other, and outputs are a function of the inputs. Functions can be non-linear or linear. If an MLP model is used for linear data, then the function should be a linear. It should have at least one input and output layer but, a multilayer perceptron can have one or more hidden layers. An input layer has no

computational role in the network; the role of the input layer is to pass an input vector to the network. A hidden layer plays a computational role, and the output layer shows the final decision as a probability. The general structure of an MLP can be seen from the figure below [66].

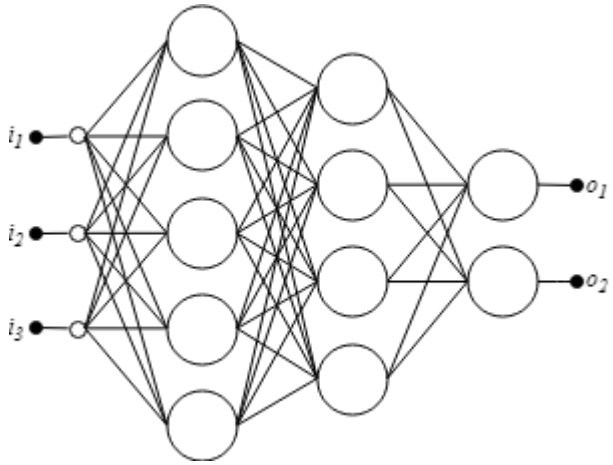

Figure 3.19: A Multilayer Perceptron with Two Hidden Layers

Figure 3.18 shows the fully connected MLP with two hidden layers. As seen from the figure, the first layer is the input vector, the middle two layers called hidden layers, and the rightmost layer is the output layer.

3.6 Feature Selection

Feature selection is the task that reduces the dimensionality of the feature space by ranking the terms according to their scores obtained with feature selection metrics and then choosing the terms with highest scores [6]. We used four different feature selection metrics in our study (CHI, MI, GSS Coefficient, and DF).

After ordering the terms as mentioned above, we modified the document vectors to consist only of these terms and performed the classification process to measure performance. We repeated this process by increasing the number of features by 1000 in each step through a loop and recorded it for evaluation.

There are generally two different types of a dictionary to apply feature selection methods:

- **Local dictionaries:** consist of class-based feature sets.
- **Global dictionaries:** consist of a single set that contains all the features in the data set.

Local dictionaries optimize the feature selection process on a class basis, whereas global dictionaries generally optimize this process [67]. Because of its ease of calculation and simplicity, we focused on the usage of general dictionaries in our thesis.

While applying feature selection methods in our study, we built a general dictionary *D* that contains all features in the dataset. We then calculated the values of *A, B, C, D* for each term according to Table 2.1, which is given in Chapter 2 and kept them in four separate lists.

After finding these values, we calculated the scores for each term by substituting the corresponding values in the formula of the feature selection method. We repeated these operations for each category *c* in the dataset. Thus, we found the scores as many as the number of categories for each word. We added the maximum score for the relevant word to the general dictionary *D*. Finally, we sorted the elements of the *D* in descending order and conducted our experiments by selecting the first *N* words from *D*.

3.7 Performance Evaluation Metrics

In our thesis, we evaluated the success of classification and feature selection methods with measurements such as Accuracy, *F*-measure, Recall and Precision [68].

3.7.1 Definitions

Accuracy: is the ratio of the number of correctly classified documents to the total number of classified documents.

Precision (P): denotes how many of the documents classified as $c_i$actually belong to the class c_i.

Recall (R): denotes how many of the documents belonging to class c_i are classified as c_i.

***F*-measure:** is the harmonic mean of precision and recall.

True Positive (TP): is the number of documents that are belonging to class c_i and classified as class c_i.

False Positive (FP): is the number of documents that are not belonging to class c_i and classified as class c_i.

True Negative (TN): is the number of documents that are not belonging to class c_i and classified as not class c_i.

False Negative (FN): is the number of documents that are belonging to class c_i and classified as not class c_i.

Macro-Average: is the average of the metrics that are calculated per category. It does not consider the class imbalance.

Micro-Average: is the average of metrics that are calculated globally by counting the total *TPs*, *FPs*, *FNs*, and *TNs*. It considers the weight of each class equally and reflects the overall performance.

Table 3.9 below illustrates the relation of *TP*, *FP*, *TN*, *FN* values in the confusion matrix.

Table 3.9: Confusion Matrix

		Predicted Class	
		c_i	$\bar{c_i}$
Actual Class	c_i	*TP*	*FN*
	$\bar{c_i}$	*FP*	*TN*

The evaluation metrics can be calculated according to the confusion matrix with following equations [57].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.17}$$

$$P = \frac{TP}{TP + FP} \tag{3.18}$$

$$R = \frac{TP}{TP + FN} \tag{3.19}$$

$$F - measure = \frac{2 \times P \times R}{P + R} \tag{3.20}$$

$$P_{micro-avg} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \tag{3.21}$$

$$R_{micro-avg} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \tag{3.22}$$

$$F - measure_{micro-avg} = \frac{2 \times P_{micro-avg} \times R_{micro-avg}}{2 + P_{micro-avg} + R_{micro-avg}} \tag{3.23}$$

$$P_{macro-avg} = \frac{\sum_{i=1}^{n} P_i}{n} \tag{3.24}$$

$$R_{macro-avg} = \frac{\sum_{i=1}^{n} R_i}{n} \tag{3.25}$$

$$F - measure_{macro-avg} = \frac{\sum_{i=1}^{n} F - measure_i}{n} \quad (3.26)$$

Where n denotes that the total number of classes in the equations (3.21), (3.22), (3.24), (3.25) and (3.26).

Chapter 4

EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Introduction

In order to evaluate the text categorization system, the experiments are conducted on four different feature selection metrics: CHI, MI, GSS and, DF. The term weighting methods such as TF-IDF, DF, probability-based term weighting is used to reflect the effect of classification algorithms comparatively.

All of the methods used in this thesis, implemented with Python programming language. The text categorization system performed on three different data sets (corpus). Precision, recall, *f*-measure, and micro-macro averaging metrics are used to evaluate the performance of all classifiers. Experiments are performed mainly on the Reuters-21578 dataset. The 4-Universities, 20 Newsgroups datasets are used to see whether the methods exhibited the same behavior.

4.2 Results for Reuters-21578 Dataset

In this section, the results of experiments conducted with Reuters-21578 dataset are presented.

4.2.1 Performance of Classification Algorithms

In this section, the performance of classification algorithms with using different term weighting schemes is given.

4.2.1.1 Using TF-IDF Weighting Scheme

The following tables provide information on the performance of different classification algorithms with the Reuters-21578 dataset in terms of precision, recall, accuracy, *f*-measure, and micro-macro averaging metrics. The TF-IDF weighting scheme is used for weighting the terms.

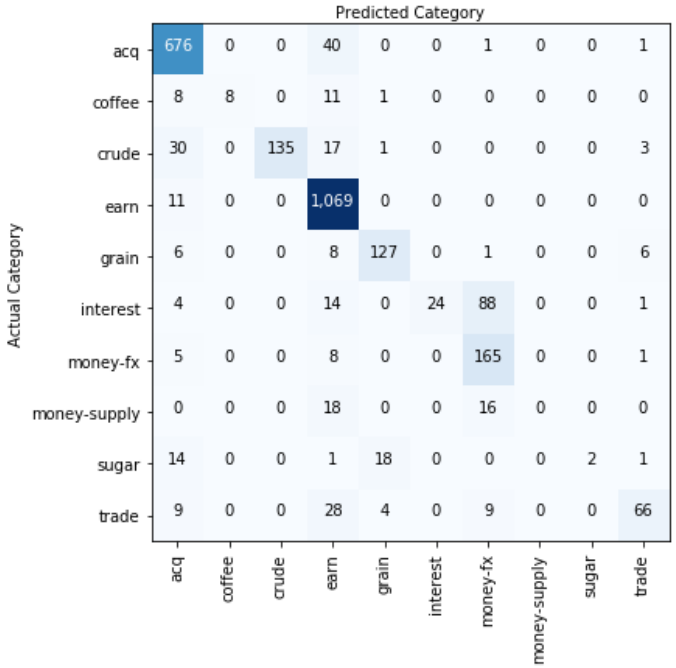

Figure 4.1: CM of MNB on Reuters-21578

The confusion matrix can be seen from Figure 4.1 above. It shows the *TP*, *TN*, *FP*, *FN* values for the classes, and the performance evaluation metric calculated by using those values with equations (3.17), (3.18), (3.19), (3.20), (3.21), (3.22), (3.23), (3.24), (3.25).

Table 4.1: CR of MNB on Reuters-21578

	Multinomial Navie Bayes Classifier			
	precision	**recall**	***f*-measure**	**support**
acq	0.8860	0.9415	0.9129	718
coffee	1.0000	0.2857	0.4444	28
crude	1.0000	0.7258	0.8411	186
earn	0.8806	0.9898	0.9320	1080
grain	0.8411	0.8581	0.8495	148
interest	1.0000	0.1832	0.3097	131
money-fx	0.5893	0.9218	0.7190	179
money-supply	0.0000	0.0000	0.0000	34
sugar	1.0000	0.0556	0.1053	36
trade	0.8354	0.5690	0.6769	116
micro-avg	0.8554	0.8554	0.8554	2656
macro-avg	0.8032	0.5530	0.5791	2656
Accuracy	0.8554			

Table 4.1 above provides the scores of performance evaluation metrics for the MNB classification algorithm. The *support* field denotes the number of positive samples for the corresponding class. The *accuracy* field indicates how many samples are classified correctly. Calculation and tabulation the values of performance evaluation metrics can be explained with the following example.

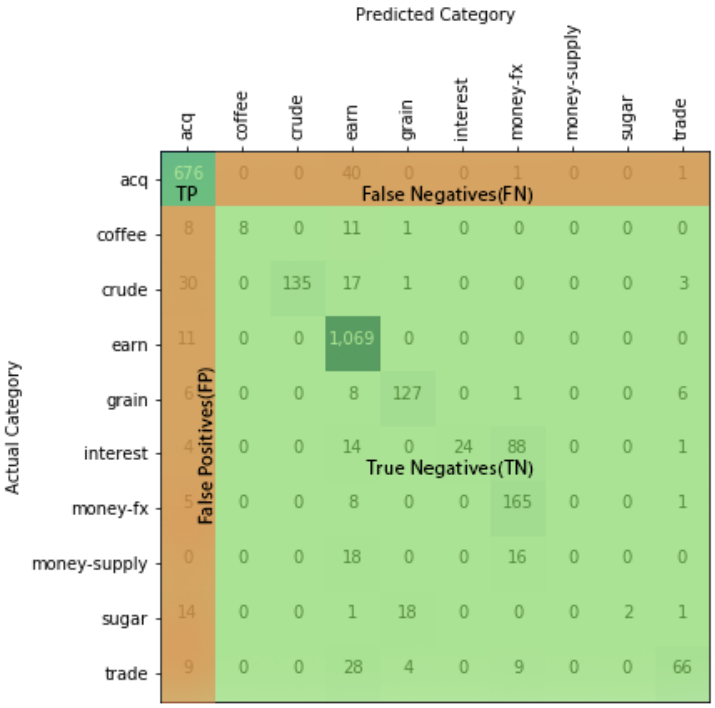

Figure 4.2: TP, TN, FP, FN Values for acq

Precision, recall, *f*-measure and accuracy scores for "acq" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.2 as in following.

$$Precision_{acq} = \frac{676}{676 + 87} = 0.8860$$

$$Recall_{acq} = \frac{676}{676 + 42} = 0.9415$$

$$f - measure_{acq} = \frac{2 \times 0.8860 \times 0.9415}{0.8860 + 0.9415} = 0.9129$$

$$Accuracy_{acq} = \frac{676 + 1851}{2656} = 0.9514$$

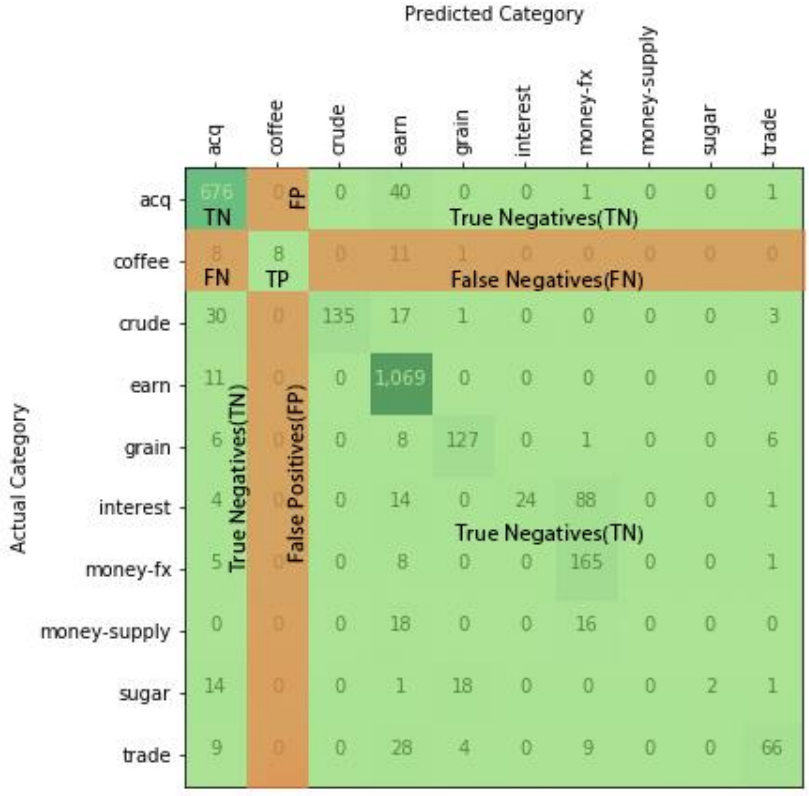

Figure 4.3: TP, TN, FP, FN Values for coffee

Precision, recall, *f*-measure and accuracy scores for "coffee" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.3 as in following.

$$Precision_{coffee} = \frac{8}{8 + 0} = 1.0000$$

$$Recall_{coffee} = \frac{8}{8 + 20} = 0.2857$$

$$f - measure_{coffee} = \frac{2 \times 1.0000 \times 0.2857}{1.0000 + 0.2857} = 0.4444$$

$$Accuracy_{coffee} = \frac{8 + 2628}{2656} = 0.9925$$

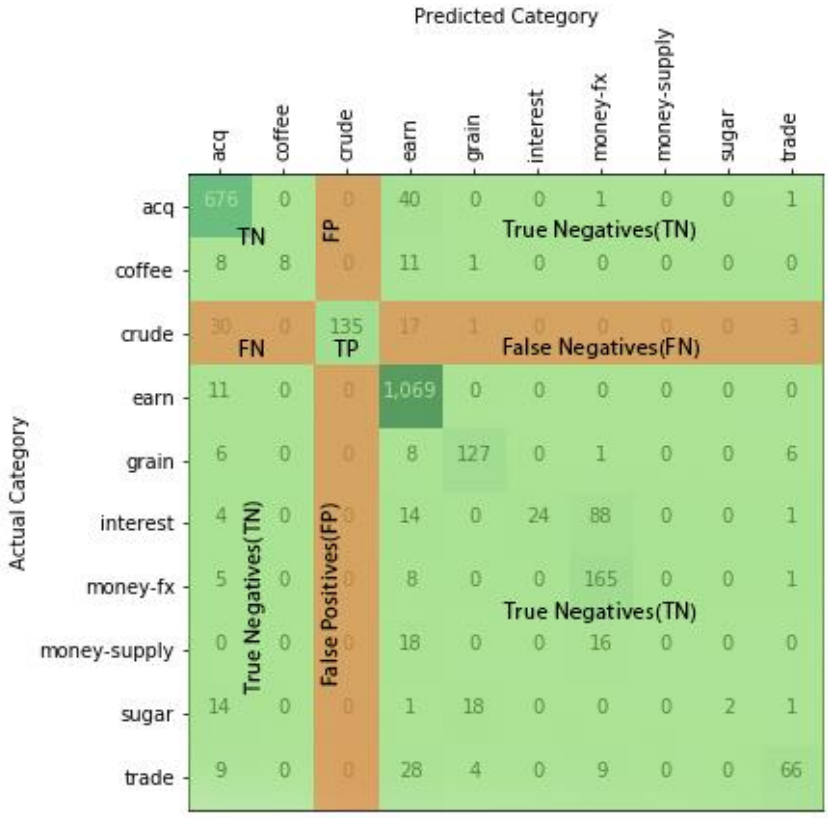

Figure 4.4: TP, TN, FP, FN Values for crude

Precision, recall, *f*-measure and accuracy scores for "crude" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.4 as in following.

$$Precision_{crude} = \frac{135}{135 + 0} = 1.0000$$

$$Recall_{crude} = \frac{135}{135 + 51} = 0.7258$$

$$f - measure_{crude} = \frac{2 \times 1.0000 \times 0.7258}{1.0000 + 0.7558} = 0.8411$$

$$Accuracy_{crude} = \frac{135 + 2470}{2656} = 0.9808$$

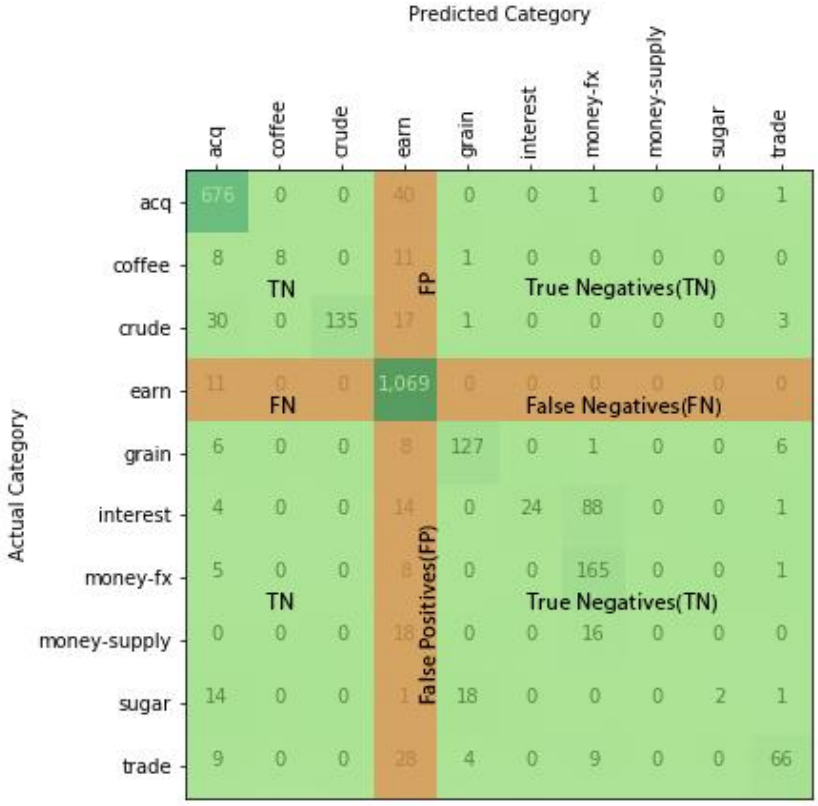

Figure 4.5: TP, TN, FP, FN Values for earn

Precision, recall, *f*-measure and accuracy scores for "earn" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.5 as in following.

$$Precision_{earn} = \frac{1069}{1069 + 145} = 0.8806$$

$$Recall_{earn} = \frac{1069}{1069 + 11} = 0.9898$$

$$f - measure_{earn} = \frac{2 \times 0.8806 \times 0.9898}{0.8806 + 0.9898} = 0.9320$$

$$Accuracy_{earn} = \frac{1069 + 1431}{2656} = 0.9413$$

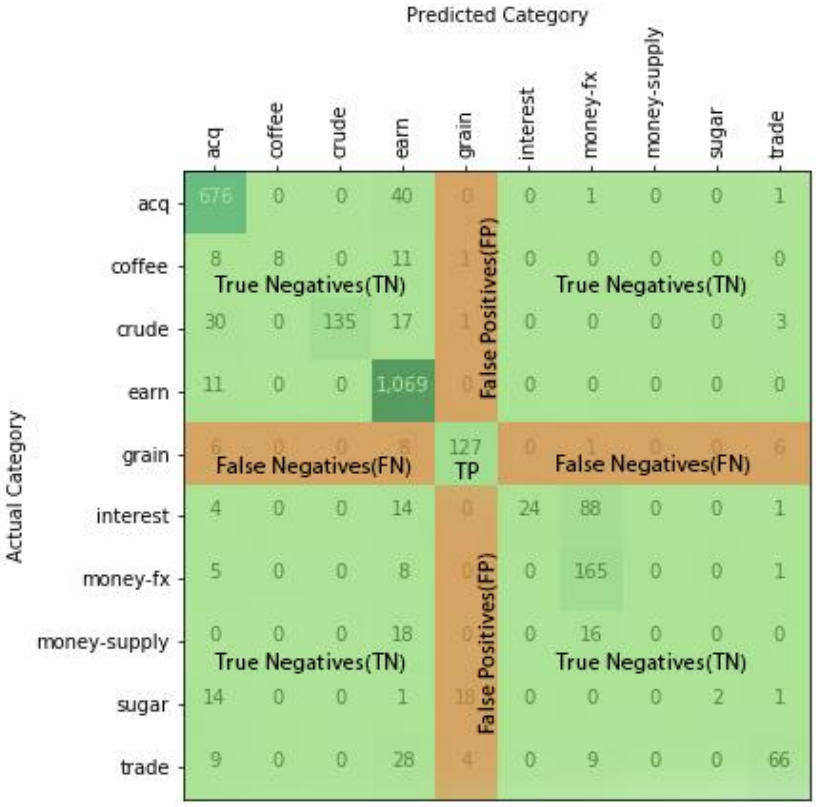

Figure 4.6: TP, TN, FP, FN Values for grain

Precision, recall, *f*-measure and accuracy scores for "grain" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.6 as in following.

$$Precision_{grain} = \frac{127}{127 + 24} = 0.8411$$

$$Recall_{grain} = \frac{127}{127 + 21} = 0.8581$$

$$f - measure_{grain} = \frac{2 \times 0.8411 \times 0.8581}{0.8411 + 0.8581} = 0.8495$$

$$Accuracy_{grain} = \frac{127 + 2484}{2656} = 0.9831$$

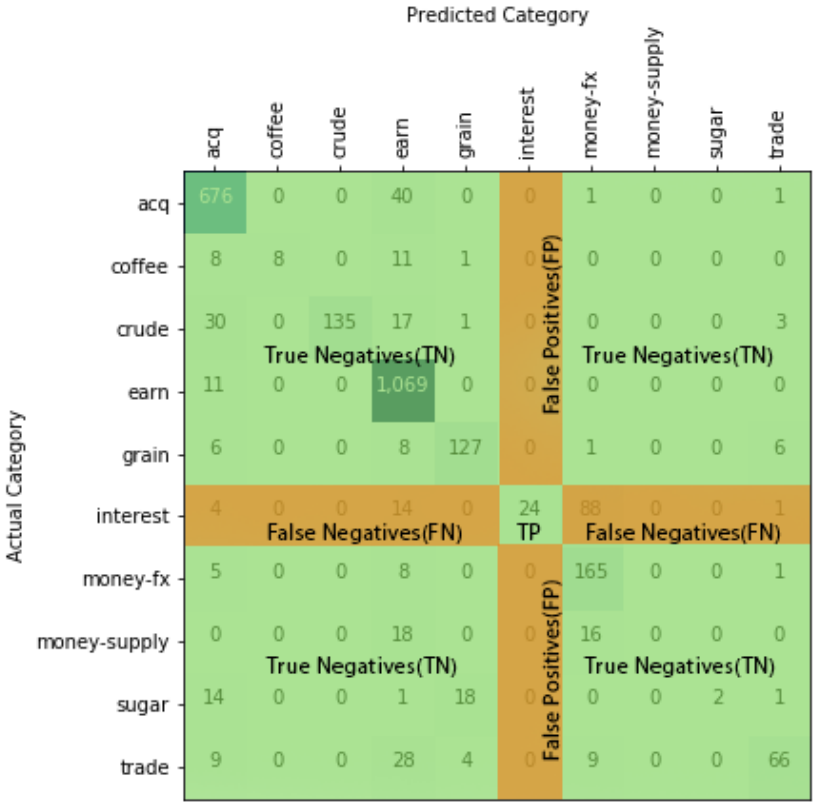

Figure 4.7: TP, TN, FP, FN Values for interest

Precision, recall, *f*-measure and accuracy scores for "interest" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.7 as in following.

$$Precision_{interest} = \frac{24}{24 + 0} = 1.0000$$

$$Recall_{interest} = \frac{24}{24 + 107} = 0.1832$$

$$f - measure_{interest} = \frac{2 \times 1.0000 \times 0.1832}{1.0000 + 0.1832} = 0.3097$$

$$Accuracy_{interest} = \frac{24 + 2525}{2656} = 0.9831$$

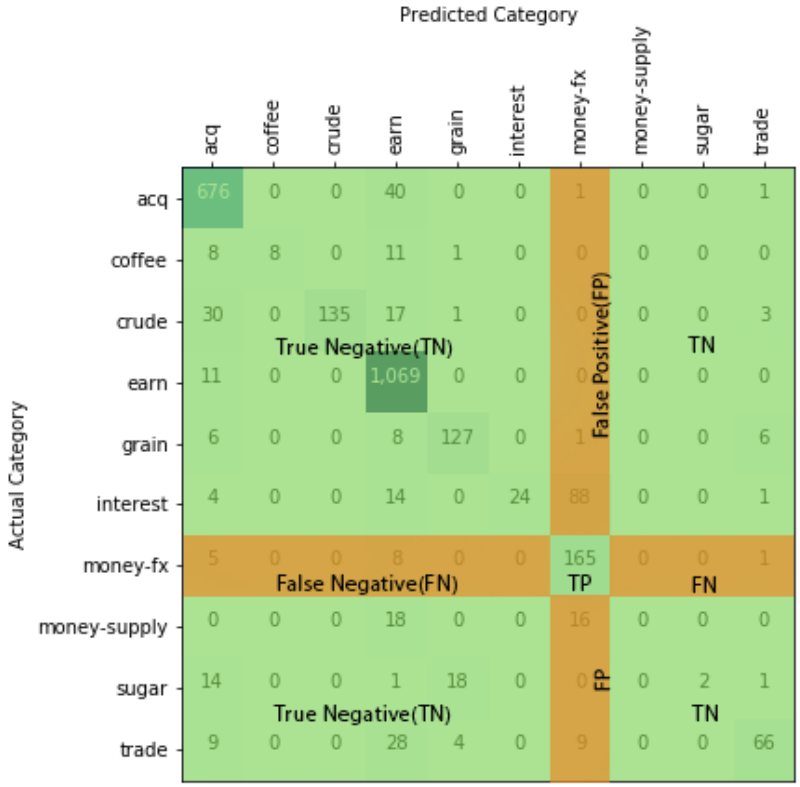

Figure 4.8: TP, TN, FP, FN Values for money-fx

Precision, recall, *f*-measure and accuracy scores for "money-fx" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.8 as in following.

$$Precision_{money-fx} = \frac{165}{165 + 115} = 0.5893$$

$$Recall_{money-fx} = \frac{165}{165 + 14} = 0.9218$$

$$f - measure_{money-fx} = \frac{2 \times 0.5893 \times 0.9218}{0.5893 + 0.9218} = 0.7190$$

$$Accuracy_{money-fx} = \frac{165 + 2362}{2656} = 0.9514$$

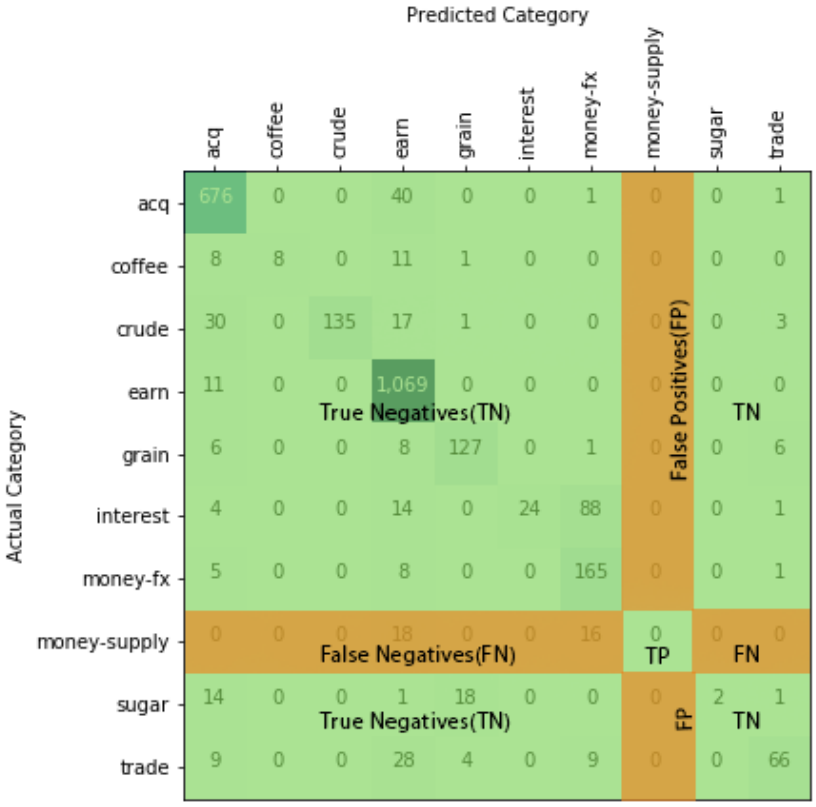

Figure 4.9: TP, TN, FP, FN Values for money-supply

Precision, recall, *f*-measure and accuracy scores for "money-supply" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.9 as in following.

$$Precision_{money-supply} = \frac{0}{0+0} = 0.0000$$

$$Recall_{money-supply} = \frac{0}{0+34} = 0.0000$$

$$f-measure_{money-supply} = \frac{2 \times 0.0000 \times 0.0000}{0.0000 + 0.0000} = 0.0000$$

$$Accuracy_{money-supply} = \frac{0+2622}{2656} = 0.9872$$

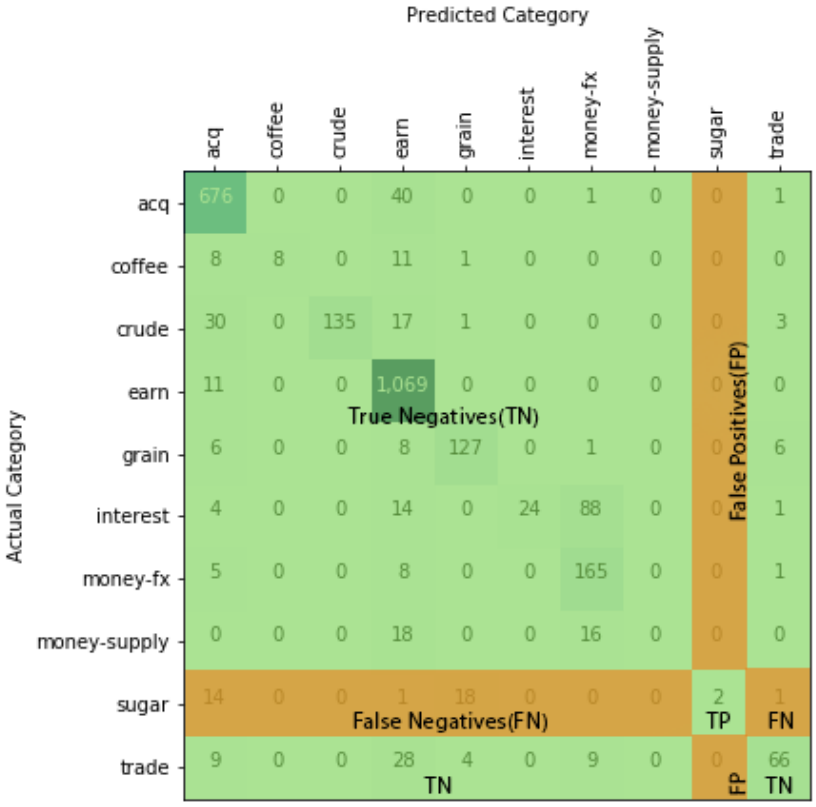

Figure 4.10: TP, TN, FP, FN Values for sugar

Precision, recall, *f*-measure and accuracy scores for "sugar" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.10 as in following.

$$Precision_{sugar} = \frac{2}{2+0} = 1.0000$$

$$Recall_{sugar} = \frac{2}{2+34} = 0.0556$$

$$f - measure_{sugar} = \frac{2 \times 1.0000 \times 0.0556}{1.0000 + 0.0556} = 0.1053$$

$$Accuracy_{sugar} = \frac{2+2620}{2656} = 0.9872$$

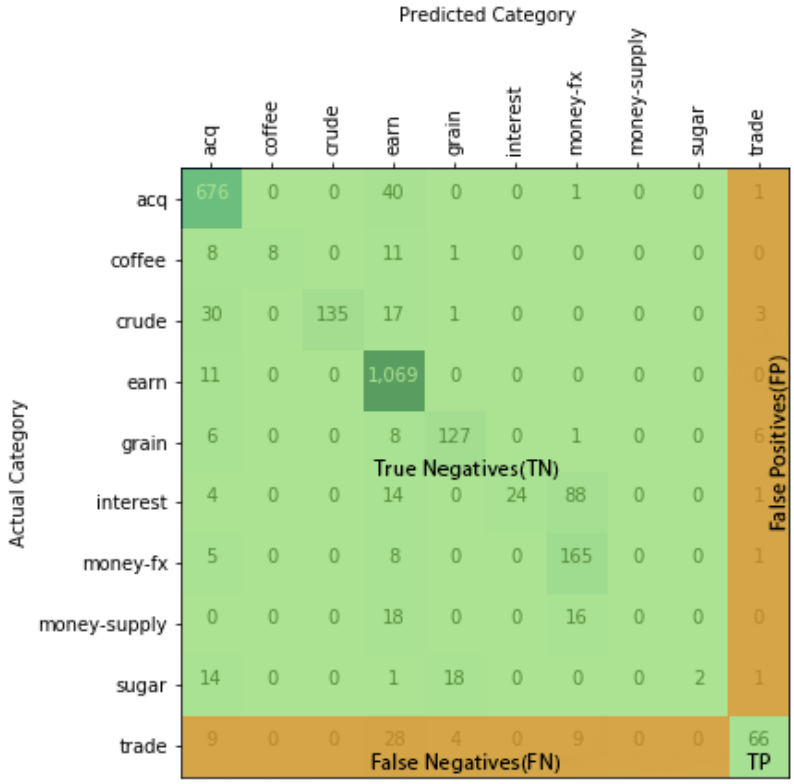

Figure 4.11: TP, TN, FP, FN Values for trade

Precision, recall, *f*-measure and accuracy scores for "trade" class can be computed by equation (3.17), (3.18), (3.19), (3.20) according to Figure 4.11 as in following.

$$Precision_{trade} = \frac{66}{66 + 13} = 0.8354$$

$$Recall_{trade} = \frac{66}{66 + 50} = 0.5690$$

$$f - measure_{trade} = \frac{2 \times 0.8354 \times 0.5690}{0.8334 + 0.5690} = 0.6769$$

$$Accuracy_{trade} = \frac{66 + 2527}{2656} = 0.9763$$

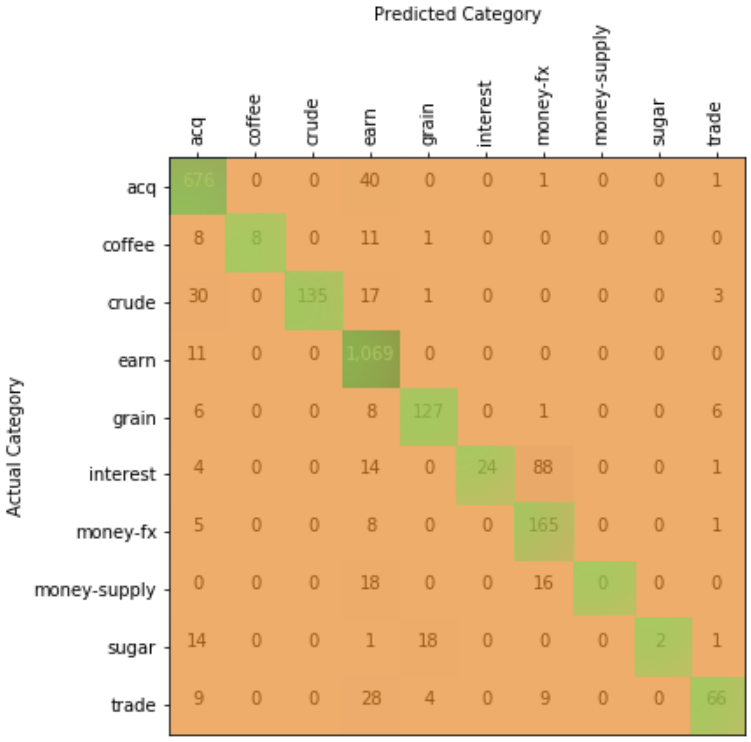

Figure 4.12: Overall Accuracy

According to Figure 4.12, the overall accuracy score of classification can be computed by dividing the number of correct predictions to the total number of test documents in the dataset as in following example.

$$Accuracy_{Overall} = \frac{676 + 8 + 135 + 1069 + 127 + 24 + 165 + 0 + 2 + 66}{2656} = 0.8554$$

Macro-average of a performance evaluation metrics can be calculated by equation (3.24), (3.25) and (3.26) as in following examples.

$$P_{macro-avg} = \frac{0.8860 + 1 + 1 + 0.8806 + 0.8411 + 1 + 0.5893 + 0 + 1 + 0.8354}{10} = 0.8032$$

$$R_{macro-avg}$$

$$= \frac{0.9415 + 0.2857 + 0.7258 + 0.9898 + 0.8581 + 0.1832 + 0.9218 + 0.0556 + 0.5690}{10}$$

$$= 0.5530$$

$$F - measure_{macro-avg}$$

$$= \frac{0.9129 + 0.4444 + 0.8411 + 0.9320 + 0.8495 + 0.3097 + 0.7190 + 0.1053 + 0.6769}{10}$$

$$= 0.8554$$

Micro-average is of performance evaluation metrics can be calculated by the sum up individual *tp*, *tn*, *fp*, and *fn* values in the confusion matrix. Therefore, as can be seen in Table 4.2, the *micro-avg* is the same value for each metric. *Micro-averaged* metrics can be calculated by equation (3.21), (3.22) and (3.23) as in following examples.

$$P_{micro-avg} = \frac{2272}{2272 + 384} = 0.8554$$

$$R_{micro-avg} = \frac{2272}{2272 + 384} = 0.8554$$

$$F - measure_{micro-avg} = \frac{2272}{2272 + 384} = 0.8554$$

Table 4.2: CR of GNB on Reuters-21578

	Gaussian Navie Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.6726	0.5780	0.6217	718
coffee	0.7000	0.7500	0.7241	28
crude	0.6455	0.7634	0.6995	186
earn	0.7741	0.8315	0.8018	1080
grain	0.7500	0.7095	0.7292	148
interest	0.5122	0.4809	0.4961	131
money-fx	0.5177	0.4078	0.4562	179
money-supply	0.4000	0.5882	0.4762	34
sugar	0.4737	0.5000	0.4865	36
trade	0.4307	0.5086	0.4664	116
micro-avg	0.6830	0.6830	0.6830	2656
macro-avg	0.5876	0.6118	0.5958	2656
Accuracy	0.6830			

Table 4.2 shows the classification details for GNB classifier. When the "support" field taken into consideration, it indicates that the Reuters-21578 dataset has unbalanced document distribution. It is easy to see that the frequent class "earn" has a high *f*-

measure value, and the documents of this category classified with 80% of success. The 68% of test documents correctly placed into the corresponding category. Confusion matrix can be seen from Figure 4.13 below.

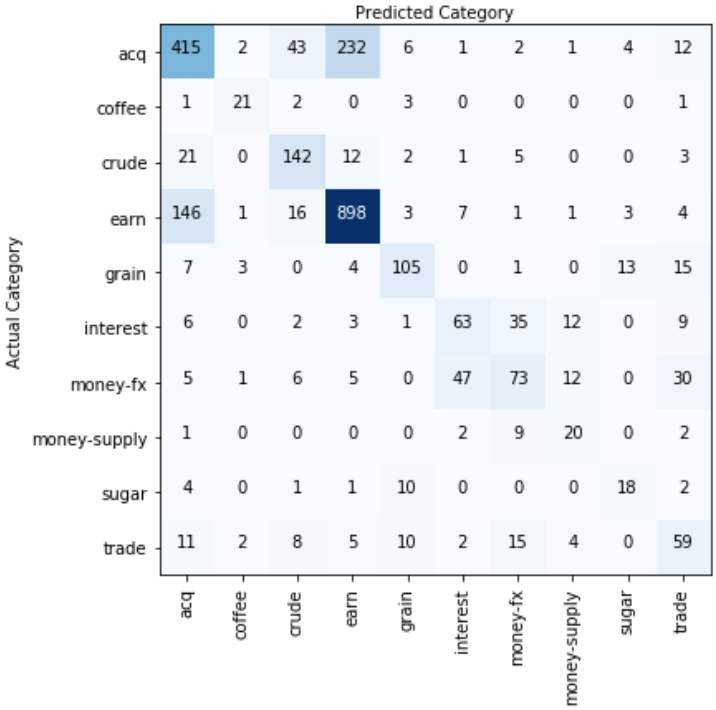

Figure 4.13: CM of GNB on Reuters-21578

Table 4.3: CR of kNN (k=1) on Reuters-21578

	kNN (*k*=1) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9267	0.6518	0.7653	718
coffee	0.8065	0.8929	0.8475	28
crude	0.8593	0.9194	0.8883	186
earn	0.8215	0.9630	0.8866	1080
grain	0.8467	0.8581	0.8523	148
interest	0.6129	0.5802	0.5961	131
money-fx	0.6865	0.7095	0.6978	179
money-supply	0.6829	0.8235	0.7467	34
sugar	0.7429	0.7222	0.7324	36
trade	0.7250	0.7500	0.7373	116
micro-avg	0.8189	0.8189	0.8189	2656
macro-avg	0.7711	0.7870	0.7750	2656
Accuracy	0.8189			

Table 4.3 above provides detailed information for classification with kNN when the *k* equals to 1. As seen from the table, kNN shows the worst performance for class "interest" and it shows the best performance for class "crude" according to *f*-measure. kNN correctly categorized the 82% of test documents. Confusion matrix can be seen from the Figure 4.14 below.

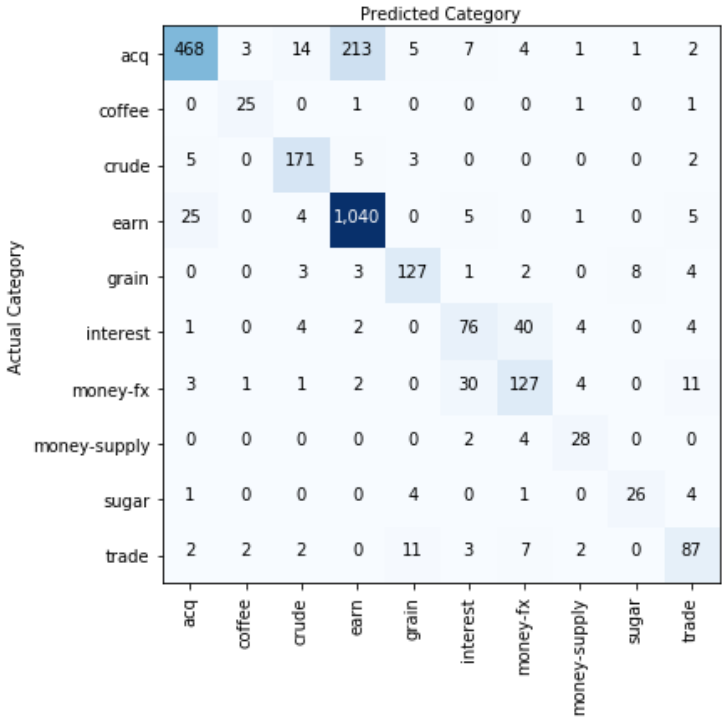

Figure 4.14: CM of kNN (*k*=1) on Reuters-21578

Table 4.4: CR of kNN (k=17) on Reuters-21578

	kNN (*k*=17) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9767	0.7591	0.8542	718
coffee	0.8065	0.8929	0.8475	28
crude	0.8800	0.9462	0.9119	186
earn	0.8769	0.9824	0.9266	1080
grain	0.9014	0.8649	0.8828	148
interest	0.7368	0.6412	0.6857	131
money-fx	0.6742	0.8324	0.7450	179
money-supply	0.7576	0.7353	0.7463	34
sugar	0.8710	0.7500	0.8060	36
trade	0.8103	0.8103	0.8103	116
micro-avg	0.8712	0.8712	0.8712	2656
macro-avg	0.8291	0.8215	0.8216	2656
Accuracy	0.8712			

Table 4.4 above, shows the classification report for kNN classifier on the Reuters-21578 dataset when the *k* selected as 17. As seen from the table, using bigger *k* value improved the classification accuracy from 82% to 87%. When the *f*-measure is taken into consideration, kNN still performs the best for category "earn". The confusion matrix of the classification process with kNN (*k*=17) can be seen from the Figure 4.15 below.

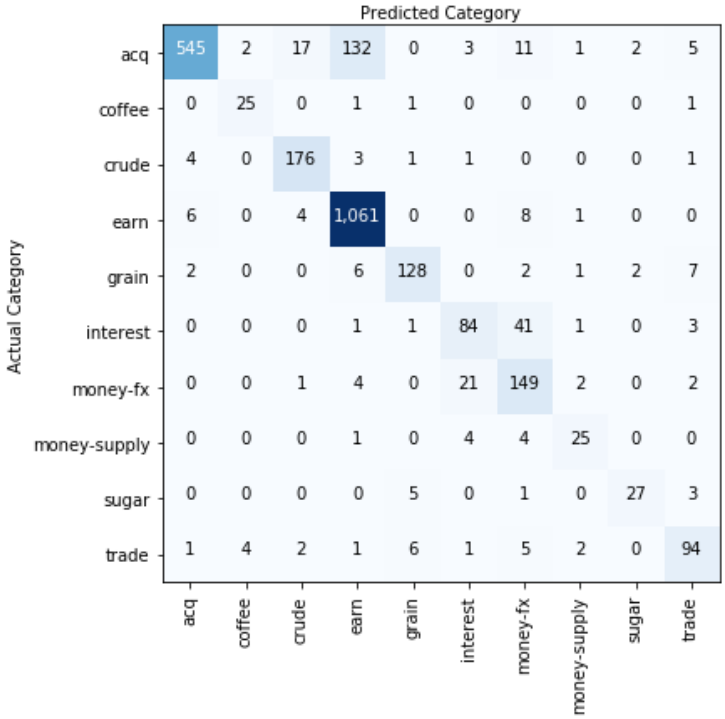

Figure 4.15: CM of kNN(*k*=17) on Reuters-21578

Table 4.5: CR of SVM (Linear Kernel) on Reuters-21578

	SVM (Linear Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9641	0.9721	0.9681	718
coffee	0.9000	0.9643	0.9310	28
crude	0.9568	0.9516	0.9542	186
earn	0.9871	0.9889	0.9880	1080
grain	0.9444	0.9189	0.9315	148
interest	0.7826	0.6870	0.7317	131
money-fx	0.7525	0.8324	0.7905	179
money-supply	0.8966	0.7647	0.8254	34
sugar	0.9333	0.7778	0.8485	36
trade	0.8571	0.8793	0.8681	116
micro-avg	0.9416	0.9416	0.9416	2656
macro-avg	0.8975	0.8737	0.8837	2656
Accuracy	0.9416			

Table 4.5 above reflects the classification details of linear SVM. It has a considerable performance on the classification, and it shows the best performance. As seen from the table, SVM correctly placed the 94% of test documents into the corresponding category. The confusion matrix can be seen from Figure 4.16 below.

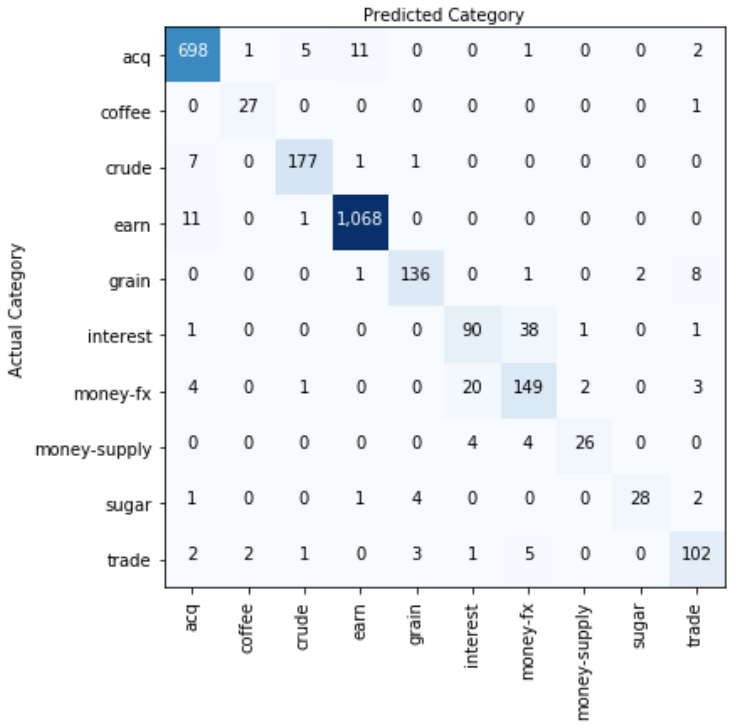

Figure 4.16: CM of SVM (Linear Kernel) on Reuters-21578

Table 4.6: CR of SVM (RBF Kernel) on Reuters-21578

	SVM (RBF Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9288	0.9805	0.9539	718
coffee	0.9259	0.8929	0.9091	28
crude	0.9500	0.9194	0.9344	186
earn	0.9925	0.9861	0.9893	1080
grain	0.9375	0.9122	0.9247	148
interest	0.7961	0.6260	0.7009	131
money-fx	0.7310	0.8045	0.7660	179
money-supply	0.8929	0.7353	0.8065	34
sugar	0.9310	0.7500	0.8308	36
trade	0.8547	0.8621	0.8584	116
micro-avg	0.9330	0.9330	0.9330	2656
macro-avg	0.8940	0.8469	0.8674	2656
Accuracy	0.9330			

Table 4.6 above shows the classification report of SVM with RBF kernel. Although the performance of SVM (RBF Kernel) is less than linear SVM, it shows better results for "interest", "money-supply" and "trade" categories. The confusion matrix of classification process can be seen from the Figure 4.17 below.

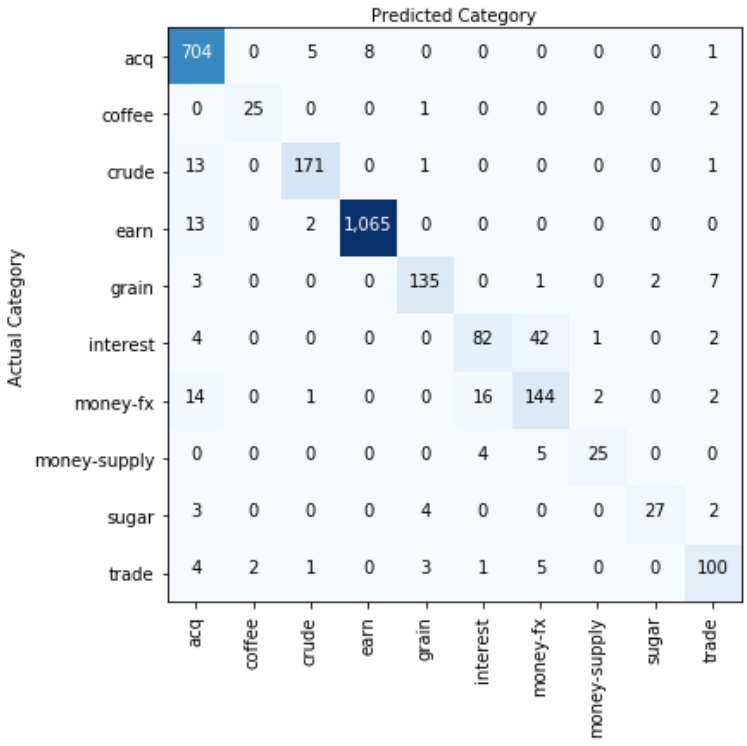

Figure 4.17: CM of SVM (RBF Kernel) on Reuters-21578

Table 4.7: CR of Decision Tree on Reuters-21578

	Decision Tree Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.8940	0.8928	0.8934	718
coffee	0.8929	0.8929	0.8929	28
crude	0.8674	0.8441	0.8556	186
earn	0.9366	0.9713	0.9536	1080
grain	0.8716	0.8716	0.8716	148
interest	0.5800	0.6641	0.6192	131
money-fx	0.6829	0.6257	0.6531	179
money-supply	0.7059	0.7059	0.7059	34
sugar	0.8750	0.5833	0.7000	36
trade	0.7556	0.5862	0.6602	116
micro-avg	0.8709	0.8709	0.8709	2656
macro-avg	0.8062	0.7638	0.7805	2656
Accuracy	0.8709			

Table 4.7 gives information on classification results with DT classifier. As seen from the table, DT shows the best performance on the classification of the documents from "earn" category and the worst performance for "interest" category. DT is correctly categorized the 87% of test documents. The confusion matrix for the classification process can be seen from the Figure 4.18 below.

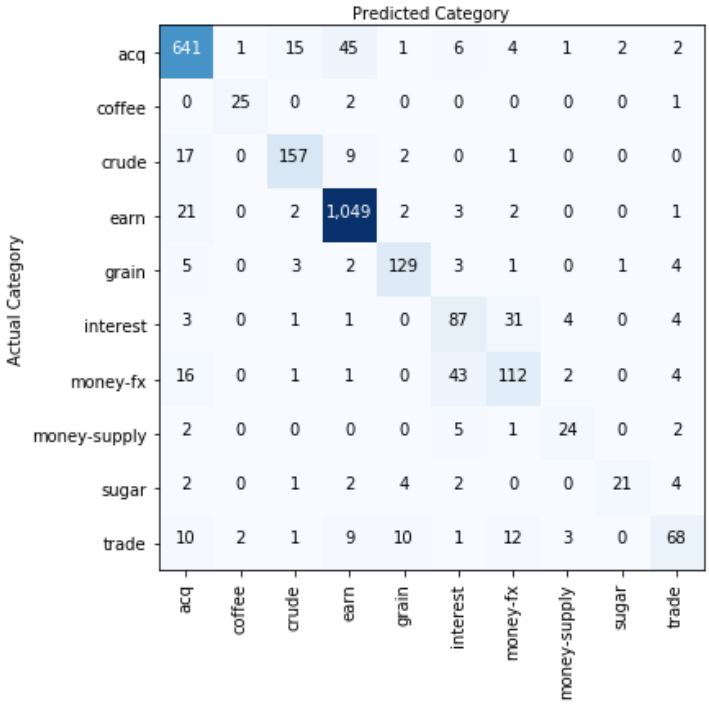

Figure 4.18: CM of DT on Reuters-21578

Table 4.8: CR of RF on Reuters-21578

	Random Forest Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.8909	0.9666	0.9272	718
coffee	0.8636	0.6786	0.7600	28
crude	0.9161	0.7634	0.8328	186
earn	0.9556	0.9769	0.9661	1080
grain	0.8299	0.8243	0.8271	148
interest	0.6809	0.4885	0.5689	131
money-fx	0.6537	0.7486	0.6979	179
money-supply	0.9048	0.5588	0.6909	34
sugar	1.0000	0.4722	0.6415	36
trade	0.7232	0.6983	0.7105	116
micro-avg	0.8837	0.8837	0.8837	2656
macro-avg	0.8419	0.7176	0.7623	2656
Accuracy	0.8837			

Table 4.8 above provides information about the classification details of RF. As seen from the table, 88% of test documents correctly placed into the corresponding category. RF shows the better performance than DT for the “acq”, “money-fx” and “trade” categories according to the *f*-measure. Figure 4.19 below shows the confusion matrix for the RF classifier.

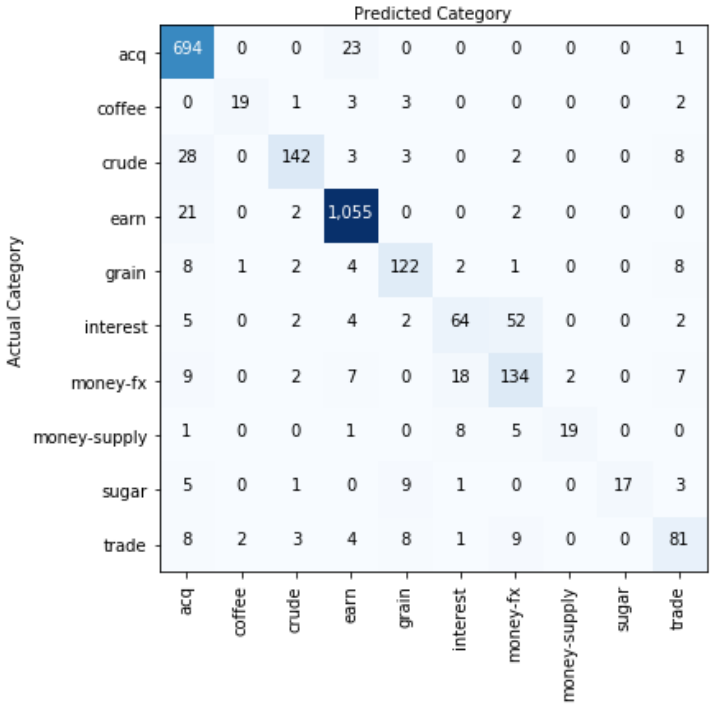

Figure 4.19: CM of RF on Reuters-21578

Table 4.9: CR of MLP on Reuters-21578

	Multilayer Perceptron (MLP) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9736	0.9749	0.9743	718
coffee	0.9615	0.8929	0.9259	28
crude	0.9516	0.9516	0.9516	186
earn	0.9889	0.9870	0.9880	1080
grain	0.9133	0.9257	0.9195	148
interest	0.7961	0.6260	0.7009	131
money-fx	0.7286	0.8547	0.7866	179
money-supply	0.8929	0.7353	0.8065	34
sugar	0.9355	0.8056	0.8657	36
trade	0.8000	0.8621	0.8299	116
micro-avg	0.9390	0.9390	0.9390	2656
macro-avg	0.8942	0.8616	0.8749	2656
Accuracy	0.9390			

Table 4.9 above gives detailed information on the performance of MLP classifier. As seen from the table, 94% of test documents correctly placed into the corresponding category. When MLP compared with other classifiers, it is the nearest classifier to linear SVM. The confusion matrix of classification process can be seen from the Figure 4.20 below.

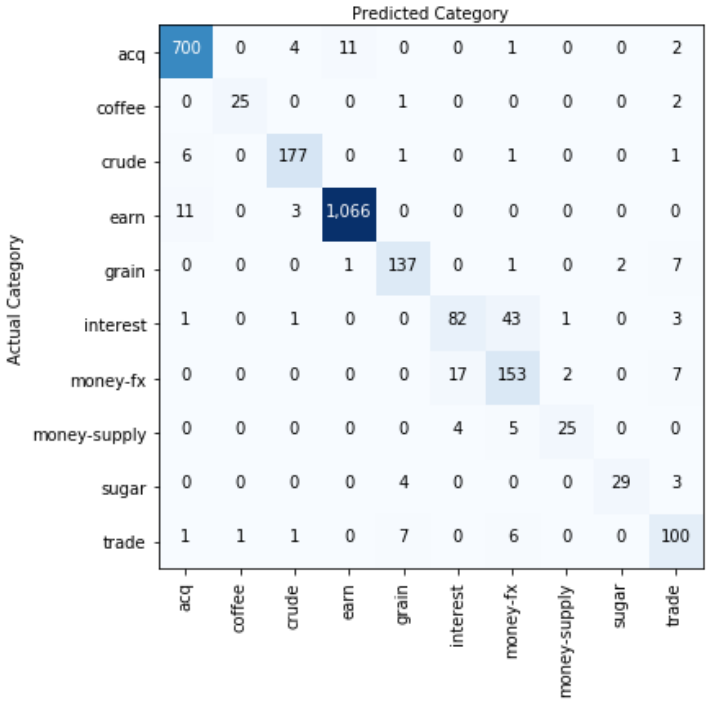

Figure 4.20: CM of MLP on Reuters-21578

Table 4.10: CR of AdaBoost on Reuters-21578

	AdaBoost Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.5383	0.9290	0.6817	718
coffee	0.8667	0.9286	0.8966	28
crude	0.7059	0.7097	0.7078	186
earn	0.9491	0.8120	0.8752	1080
grain	0.9203	0.8581	0.8881	148
interest	0.0000	0.0000	0.0000	131
money-fx	0.0000	0.0000	0.0000	179
money-supply	0.0000	0.0000	0.0000	34
sugar	0.8182	0.7500	0.7826	36
trade	0.6190	0.5603	0.5882	116
micro-avg	0.7233	0.7233	0.7233	2656
macro-avg	0.5418	0.5548	0.5420	2656
Accuracy	0.7233			

Table 4.10 above, shows the values of performance evaluation metrics for AdaBoost classifier. As seen from the table, AdaBoost is insufficient to categorize the documents of "interest", "money-fx" and "money-supply" categories. On the other hand, it shows good performance of the infrequent category "coffee". The confusion matrix of classification process can be seen from the Figure 4.21 below.

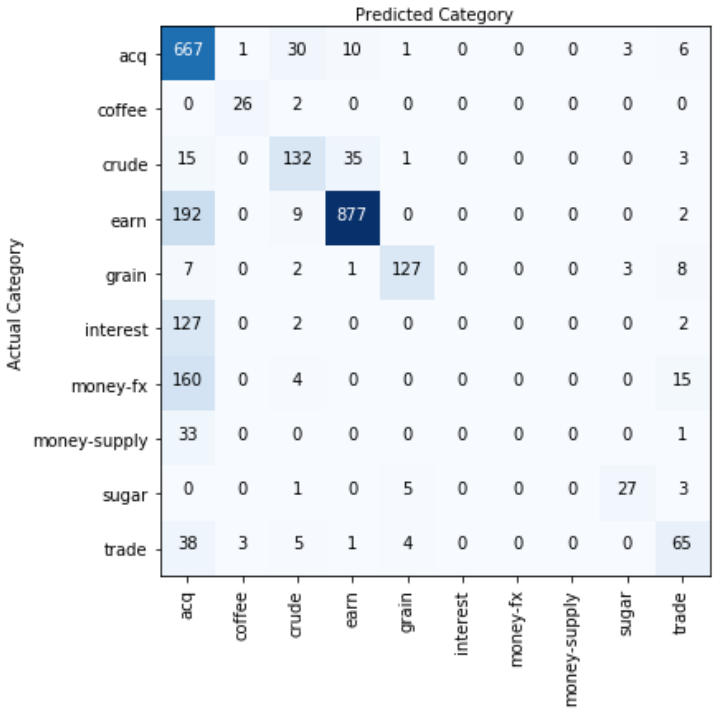

Figure 4.21: CM of AdaBoost on Reuters-21578

4.2.1.2 Using DF Weighting Scheme

The following tables give detailed information on the performance of different classification algorithms with the Reuters-21578 dataset in terms of precision, recall, accuracy, *f*-measure, and micro-macro averaging metrics. The DF weighting scheme used for weighting the terms.

Table 4.11: CR of MNB with DF Weighting Scheme

	Multinomial Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.8796	0.7326	0.7994	718
coffee	0.3462	0.3214	0.3333	28
crude	0.5155	0.5376	0.5263	186
earn	0.7941	0.9176	0.8514	1080
grain	0.8021	0.5203	0.6311	148
interest	0.3860	0.3359	0.3592	131
money-fx	0.5154	0.3743	0.4337	179
money-supply	0.4062	0.3824	0.3939	34
sugar	0.2857	0.5000	0.3636	36
trade	0.4387	0.5862	0.5018	116
micro-avg	0.7203	0.7203	0.7203	2656
macro-avg	0.5369	0.5208	0.5194	2656
Accuracy	0.7203			

Table 4.11 above, shows the classification details with DF term weighting scheme.

Table 4.12: CR of GNB with DF Weighting Scheme

	Gaussian Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.7953	0.5195	0.6285	718
coffee	0.1935	0.2143	0.2034	28
crude	0.6296	0.5484	0.5862	186
earn	0.8031	0.8537	0.8276	1080
grain	0.6014	0.5608	0.5804	148
interest	0.2535	0.2748	0.2637	131
money-fx	0.4436	0.3296	0.3782	179
money-supply	0.0769	0.6176	0.1368	34
sugar	0.1429	0.1944	0.1647	36
trade	0.4144	0.3966	0.4053	116
micro-avg	0.6231	0.6231	0.6231	2656
macro-avg	0.4354	0.4510	0.4175	2656
Accuracy	0.6231			

Table 4.12 above, provides the detailed classification results of GNB classifier with DF weighting scheme.

Table 4.13: CR of kNN (k=1) with DF Weighting Scheme

	kNN (*k*=1) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.5412	0.6407	0.5867	718
coffee	0.1429	0.1071	0.1224	28
crude	0.4026	0.1667	0.2357	186
earn	0.7850	0.8556	0.8188	1080
grain	0.4085	0.3919	0.4000	148
interest	0.2089	0.2519	0.2284	131
money-fx	0.2039	0.1173	0.1489	179
money-supply	0.3333	0.2059	0.2545	34
sugar	0.1395	0.1667	0.1519	36
trade	0.2969	0.1638	0.2111	116
micro-avg	0.5881	0.5881	0.5881	2656
macro-avg	0.3463	0.3067	0.3159	2656
Accuracy	0.5881			

Table 4.13 above, gives the classification results of kNN classifier with DF weighting scheme when the *k* equals to 1.

Table 4.14: CR of kNN (k=17) with DF Weighting Scheme

	kNN (*k*=17) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.5447	0.6964	0.6112	718
coffee	0.0000	0.0000	0.0000	28
crude	0.5455	0.0968	0.1644	186
earn	0.7071	0.8630	0.7773	1080
grain	0.3264	0.3176	0.3219	148
interest	0.1864	0.1679	0.1767	131
money-fx	0.3191	0.1676	0.2198	179
money-supply	1.0000	0.0588	0.1111	34
sugar	0.0000	0.0000	0.0000	36
trade	0.1852	0.0431	0.0699	116
micro-avg	0.5858	0.5858	0.5858	2656
macro-avg	0.3814	0.2411	0.2452	2656
Accuracy	0.5858433734939759			

Table 4.14 above, shows the classification result of kNN classifier with *k*=17 when the DF weighting scheme used.

Table 4.15: CR of SVM (Linear Kernel) with DF Weighting Scheme

	SVM (Linear Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.7297	0.9025	0.8070	718
coffee	0.1864	0.3929	0.2529	28
crude	0.6639	0.4355	0.5260	186
earn	0.9288	0.9185	0.9236	1080
grain	0.6525	0.6216	0.6367	148
interest	0.3793	0.2519	0.3028	131
money-fx	0.4931	0.3966	0.4396	179
money-supply	0.4545	0.2941	0.3571	34
sugar	0.3000	0.3333	0.3158	36
trade	0.4824	0.3534	0.4080	116
micro-avg	0.7496	0.7496	0.7496	2656
macro-avg	0.5271	0.4900	0.4969	2656
Accuracy	0.7496			

Table 4.15 above, gives the classification result of SVM (Linear Kernel) when the DF weighting used.

Table 4.16: CR of SVM (RBF Kernel) with DF Weighting Scheme

	SVM (RBF Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.3178	0.1045	0.1572	718
coffee	0.0000	0.0000	0.0000	28
crude	0.0000	0.0000	0.0000	186
earn	0.4341	0.9667	0.5991	1080
grain	1.0000	0.0270	0.0526	148
interest	1.0000	0.0458	0.0876	131
money-fx	0.6667	0.0112	0.0220	179
money-supply	1.0000	0.0294	0.0571	34
sugar	0.0000	0.0000	0.0000	36
trade	1.0000	0.0086	0.0171	116
micro-avg	0.4266	0.4266	0.4266	2656
macro-avg	0.5419	0.1193	0.0993	2656
Accuracy	0.4266			

Table 4.16 above, shows the classification report of SVM (RBF Kernel) with DF weighting scheme.

Table 4.17: CR of Decision Tree with DF Weighting Scheme

	Decision Tree Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.6572	0.8036	0.7231	718
coffee	0.2692	0.2500	0.2593	28
crude	0.4917	0.3172	0.3856	186
earn	0.8770	0.8981	0.8875	1080
grain	0.6288	0.5608	0.5929	148
interest	0.2837	0.3053	0.2941	131
money-fx	0.4419	0.2123	0.2868	179
money-supply	0.3929	0.3235	0.3548	34
sugar	0.1935	0.1667	0.1791	36
trade	0.3981	0.3707	0.3839	116
micro-avg	0.6905	0.6905	0.6905	2656
macro-avg	0.4634	0.4208	0.4347	2656
Accuracy	0.6905			

Table 4.17 above, provides the classification details of DT classifier with DF weighting.

Table 4.18: CR of Random Forest with DF Weighting Scheme

	Random Forest Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.6204	0.9401	0.7475	718
coffee	0.8000	0.1429	0.2424	28
crude	0.6897	0.2151	0.3279	186
earn	0.8658	0.9259	0.8949	1080
grain	0.6636	0.4932	0.5659	148
interest	0.4706	0.1832	0.2637	131
money-fx	0.4950	0.2793	0.3571	179
money-supply	0.7500	0.2647	0.3913	34
sugar	0.5000	0.0556	0.1000	36
trade	0.4722	0.2931	0.3617	116
micro-avg	0.7195	0.7195	0.7195	2656
macro-avg	0.6327	0.3793	0.4252	2656
Accuracy	0.7195			

Table 4.18 above, reflects the classification details of RF classifier with DF weighting scheme.

Table 4.19: CR of MLP with DF Weighting Scheme

	Multilayer Perceptron (MLP) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.7030	0.9067	0.7920	718
coffee	0.3448	0.3571	0.3509	28
crude	0.6389	0.3710	0.4694	186
earn	0.9000	0.9167	0.9083	1080
grain	0.6296	0.5743	0.6007	148
interest	0.4500	0.2061	0.2827	131
money-fx	0.5135	0.4246	0.4648	179
money-supply	0.4118	0.2059	0.2745	34
sugar	0.2333	0.1944	0.2121	36
trade	0.4078	0.3621	0.3836	116
micro-avg	0.7395	0.7395	0.7395	2656
macro-avg	0.5233	0.4519	0.4739	2656
Accuracy	0.7395			

Table 4.19 above, shows the classification report for MLP classifier with DF weighting scheme.

Table 4.20: CR of AdaBoost with DF Weighting Scheme

	AdaBoost Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.5252	0.8858	0.6594	718
coffee	0.6667	0.1429	0.2353	28
crude	0.6111	0.2366	0.3411	186
earn	0.9312	0.8398	0.8832	1080
grain	0.7273	0.5405	0.6202	148
interest	0.0000	0.0000	0.0000	131
money-fx	0.3333	0.2961	0.3136	179
money-supply	0.1667	0.0294	0.0500	34
sugar	0.3500	0.1944	0.2500	36
trade	0.3505	0.2931	0.3192	116
micro-avg	0.6649	0.6649	0.6649	2656
macro-avg	0.4662	0.3459	0.3672	2656
Accuracy	0.6649			

Table 4.20 above, gives the classification details of AdaBoost classifier when the DF used as term weighting method.

4.2.1.3 Using Probability-based Term Weighting Approach

As mentioned chapter 3, Liu, Y., Loh, H. T., & Sun, A. [34], proposed a probability-based term weighting approach for imbalanced datasets. As can be seen from Figures 3.1 and 3.2, Reuters-21578 dataset is highly skewed. Therefore, we used the proposed method.

The tables below provide classification reports of different classifiers for probability-based term weighting.

Table 4.21: CR of MNB with Probability-based Term Weighting

	Multinomial Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9674	0.9930	0.9801	718
coffee	1.0000	0.8929	0.9434	28
crude	1.0000	0.9785	0.9891	186
earn	0.9408	1.0000	0.9695	1080
grain	1.0000	0.9730	0.9863	148
interest	1.0000	0.7099	0.8304	131
money-fx	0.8778	0.8827	0.8802	179
money-supply	0.0000	0.0000	0.0000	34
sugar	1.0000	0.9167	0.9565	36
trade	1.0000	0.9828	0.9913	116
micro-avg	0.9571	0.9571	0.9571	2656
macro-avg	0.8786	0.8329	0.8527	2656
Accuracy	0.9571			

Table 4.21 above, provides the classification report for MNB classifier with probability-based term weighting approach.

Table 4.22: CR of GNB with Probability-based Term Weighting

	Gaussian Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	1.0000	0.9986	0.9993	718
coffee	1.0000	1.0000	1.0000	28
crude	1.0000	1.0000	1.0000	186
earn	1.0000	1.0000	1.0000	1080
grain	0.9933	1.0000	0.9966	148
interest	0.8912	1.0000	0.9424	131
money-fx	1.0000	0.9050	0.9501	179
money-supply	0.9714	1.0000	0.9855	34
sugar	1.0000	1.0000	1.0000	36
trade	1.0000	1.0000	1.0000	116
micro-avg	0.9932	0.9932	0.9932	2656
macro-avg	0.9856	0.9904	0.9874	2656
Accuracy	0.9932			

Table 4.22 above, shows the classification details for GNB classifier with probability based term weighting.

Table 4.23: CR of kNN (k=1) with Probability-based Term Weighting

	kNN (*k*=1) Classifier			
	precision	**recall**	**f-measure**	**Support**
acq	0.9986	0.9652	0.9816	718
coffee	1.0000	1.0000	1.0000	28
crude	1.0000	0.9785	0.9891	186
earn	0.9773	0.9981	0.9876	1080
grain	1.0000	0.9730	0.9863	148
interest	0.8971	0.9313	0.9139	131
money-fx	0.9492	0.9385	0.9438	179
money-supply	0.8684	0.9706	0.9167	34
sugar	0.9722	0.9722	0.9722	36
trade	0.9831	1.0000	0.9915	116
micro-avg	0.9785	0.9785	0.9785	2656
macro-avg	0.9646	0.9727	0.9683	2656
Accuracy	0.9785			

Table 4.23 above, provides classification details of kNN (*k*=1) classifier with probability-based term weighting.

Table 4.24: CR of kNN (k=17) with Probability-based Term Weighting

	kNN (*k*=17) Classifier			
	precision	**recall**	**f-measure**	**Support**
acq	0.9955	0.9304	0.9618	718
coffee	0.9655	1.0000	0.9825	28
crude	1.0000	0.9785	0.9891	186
earn	1.0000	0.9944	0.9972	1080
grain	1.0000	0.9189	0.9577	148
interest	0.8779	0.8779	0.8779	131
money-fx	0.9112	0.8603	0.8851	179
money-supply	0.2783	0.9412	0.4295	34
sugar	1.0000	0.9722	0.9859	36
trade	0.9912	0.9741	0.9826	116
micro-avg	0.9552	0.9552	0.9552	2656
macro-avg	0.9020	0.9448	0.9049	2656
Accuracy	0.9552			

Table 4.24 above, shows the classification report of kNN (*k*=17) classifier with probability-based term weighting.

Table 4.25: CR of SVM (Linear Kernel) with Probability-based Term Weighting

	SVM (Linear Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9410	1.0000	0.9696	718
coffee	1.0000	0.9286	0.9630	28
crude	1.0000	0.9785	0.9891	186
earn	1.0000	0.9944	0.9972	1080
grain	1.0000	0.9392	0.9686	148
interest	0.9911	0.8473	0.9136	131
money-fx	0.9266	0.9162	0.9213	179
money-supply	0.8889	0.9412	0.9143	34
sugar	1.0000	0.9444	0.9714	36
trade	1.0000	0.9741	0.9869	116
micro-avg	0.9763	0.9763	0.9763	2656
macro-avg	0.9748	0.9464	0.9595	2656
Accuracy	0.9763			

Table 4.25 above, reflects the classification result of SVM (Linear Kernel) with probability-based term weighting.

Table 4.26: CR of SVM (RBF Kernel) with Probability-based Term Weighting

	SVM (RBF Kernel) Classifier			
	precision	**recall**	**f-measure**	**Support**
acq	0.9660	0.9889	0.9773	718
coffee	1.0000	0.9286	0.9630	28
crude	1.0000	0.8763	0.9341	186
earn	0.9746	0.9954	0.9849	1080
grain	1.0000	0.9459	0.9722	148
interest	0.9732	0.8321	0.8971	131
money-fx	0.9326	0.9274	0.9300	179
money-supply	0.6939	1.0000	0.8193	34
sugar	1.0000	0.9167	0.9565	36
trade	0.9829	0.9914	0.9871	116
micro-avg	0.9680	0.9680	0.9680	2656
macro-avg	0.9523	0.9403	0.9421	2656
Accuracy	0.9680			

Table 4.26 above, depicts the performance of SVM (RBF Kernel) with probability-based term weighting.

Table 4.27: CR of DT with Probability-based Term Weighting

	Decision Tree Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9782	0.9986	0.9883	718
coffee	1.0000	1.0000	1.0000	28
crude	1.0000	0.9785	0.9891	186
earn	0.9981	0.9954	0.9968	1080
grain	0.9586	0.9392	0.9488	148
interest	0.9829	0.8779	0.9274	131
money-fx	0.8973	0.9274	0.9121	179
money-supply	0.8333	0.8824	0.8571	34
sugar	1.0000	1.0000	1.0000	36
trade	0.9573	0.9655	0.9614	116
micro-avg	0.9789	0.9789	0.9789	2656
macro-avg	0.9606	0.9565	0.9581	2656
Accuracy	0.9789			

Table 4.27 above, gives the classification details of DT classifier with probability-based term weighting.

Table 4.28: CR of RF with Probability-based Term Weighting

	Random Forest Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9769	1.0000	0.9883	718
coffee	1.0000	0.9643	0.9818	28
crude	1.0000	0.9892	0.9946	186
earn	0.9991	0.9981	0.9986	1080
grain	0.9932	0.9865	0.9898	148
interest	0.9697	0.9771	0.9734	131
money-fx	0.9883	0.9441	0.9657	179
money-supply	1.0000	0.9118	0.9538	34
sugar	1.0000	0.9722	0.9859	36
trade	0.9913	0.9828	0.9870	116
micro-avg	0.9902	0.9902	0.9902	2656
macro-avg	0.9918	0.9726	0.9819	2656
Accuracy	0.9902			

Table 4.28 above, gives the classification details of RF classifier with probability-based term weighting.

Table 4.29: CR of MLP with Probability-based Term Weighting

	Multilayer Perceptron (MLP) Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.9986	0.9958	0.9972	718
coffee	1.0000	1.0000	1.0000	28
crude	1.0000	0.9946	0.9973	186
earn	0.9991	0.9991	0.9991	1080
grain	1.0000	0.9932	0.9966	148
interest	0.9618	0.9618	0.9618	131
money-fx	0.9725	0.9888	0.9806	179
money-supply	0.9706	0.9706	0.9706	34
sugar	1.0000	1.0000	1.0000	36
trade	0.9915	1.0000	0.9957	116
micro-avg	0.9947	0.9947	0.9947	2656
macro-avg	0.9894	0.9904	0.9899	2656
Accuracy	0.9947			

Table 4.29 above, provides the classification details of MLP classifier with probability-based term weighting.

Table 4.30: CR of AdaBoost with Probability-based Term Weighting

	AdaBoost Classifier			
	precision	**recall**	***f*-measure**	**Support**
acq	0.6575	1.0000	0.7934	718
coffee	1.0000	1.0000	1.0000	28
crude	1.0000	0.9785	0.9891	186
earn	1.0000	0.9370	0.9675	1080
grain	1.0000	0.0135	0.0267	148
interest	1.0000	0.4809	0.6495	131
money-fx	0.8632	0.4581	0.5985	179
money-supply	0.9032	0.8235	0.8615	34
sugar	1.0000	0.9722	0.9859	36
trade	0.9569	0.9569	0.9569	116
micro-avg	0.8513	0.8513	0.8513	2656
macro-avg	0.9381	0.7621	0.7829	2656
Accuracy	0.8513			

Table 4.30 above, depicts the classification results of AdaBoost classifier with probability-based term weighting.

As can be seen from the tables above, SVM (Linear Kernel) shows the highest performance when using TF-IDF weighting method for Reuters-21578 dataset.

When DF weighting used in our experiments, SVM still performs best despite a significant decrease in performance compared to TF-IDF weighting.

As a result of the experiments performed by using probability-based term weighting method, a significant increase observed on the performance of all classification algorithms. GNB performs the best in this method.

As a result of experimental observations, it is observed that TF-IDF weighting method is better than DF weighting method and probability-based weighting method has a positive effect on performance in imbalanced datasets.

As stated in the Sebastiani F.'s survey [12], SVMs exhibit best on text data, DTs perform close to SVMs, and probabilistic algorithms such as NB show worst performance. Besides, NNs such as MLP gives good results on text data.

4.2.2 Performance of Feature Selection Methods

In this section, the results of the experiments on the performance of the feature selection methods demonstrated through graphs.

4.2.2.1 Results Based on Feature Selection Metrics

In this section, we provide the experimental results of assessing the performance of the feature selection metrics with different classifiers. The aim is to investigate and compare the differences between classification algorithms when a specific feature selection method used.

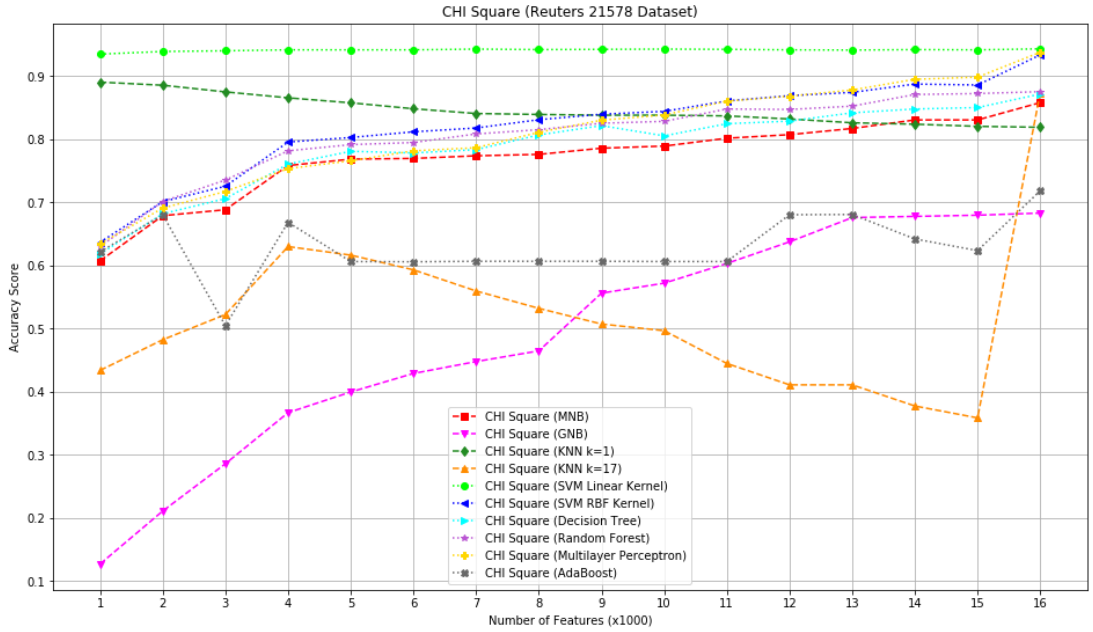

Figure 4.22: Performance of CHI on Reuters-21578 Dataset

Figure 4.22 shows the experimental results of different classification algorithms when the CHI employed as a feature selection metric. CHI is experimented by varying size of feature subsets in a range from 1000 to 16000.

As can be seen from the Figure 4.22, the results show that the MLP and SVM with RBF kernel show similar behavior. The linear SVM shows the best performance, while GNB shows the worst performance when number of features less than 9000. Also, DT and RF classifiers are show a similar action. When the two kNN classifiers compared, the results indicate that kNN (k=1) performs the best after linear SVM and choosing of larger k value leads to a decrease in terms of classification accuracy. Although AdaBoost performs more than 60 percent in general, it performs better as the size of the feature increases.

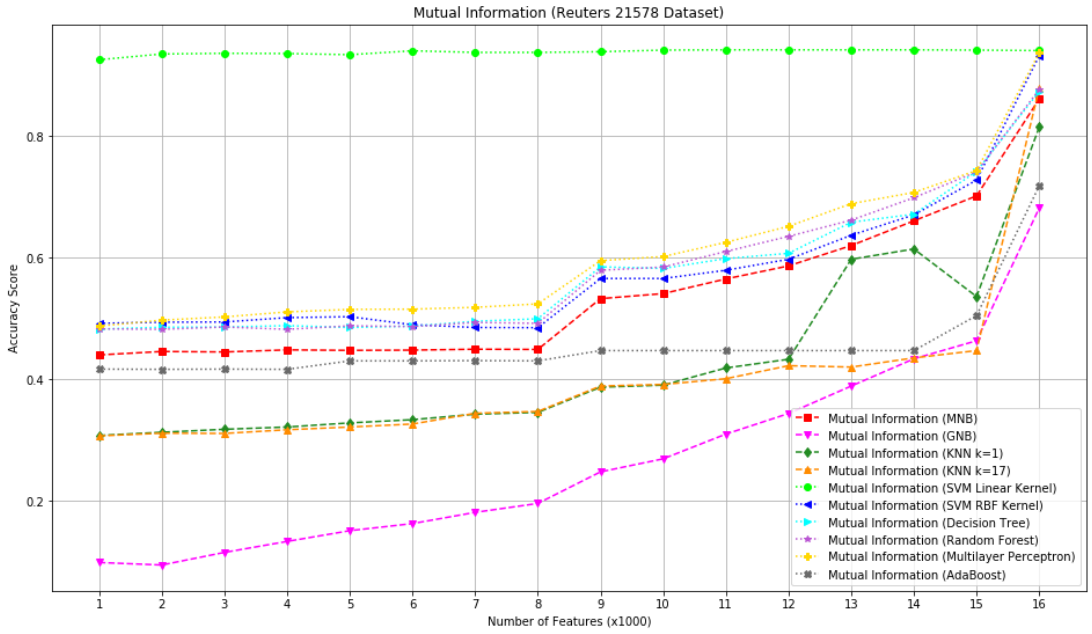

Figure 4.23: Performance of MI on Reuters-21578

Figure 4.23 above provides the experimental results of different classification algorithms when the MI is used for feature selection. MI is employed with choosing various subsets from the feature space in a range of 1000 and 16000.

As seen from the Figure 4.23, the linear SVM shows the best performance. The behavior of both kNN classifiers is almost the same while the kNN(k=1) works better

in the interval of 13000-14000. MLP shows the best performance after the linear SVM. The achievements of MNB, SVM (RBF Kernel), RF, DT, MLP, and AdaBoost classifiers are increased after using 8000 features. GNB shows the worst performance.

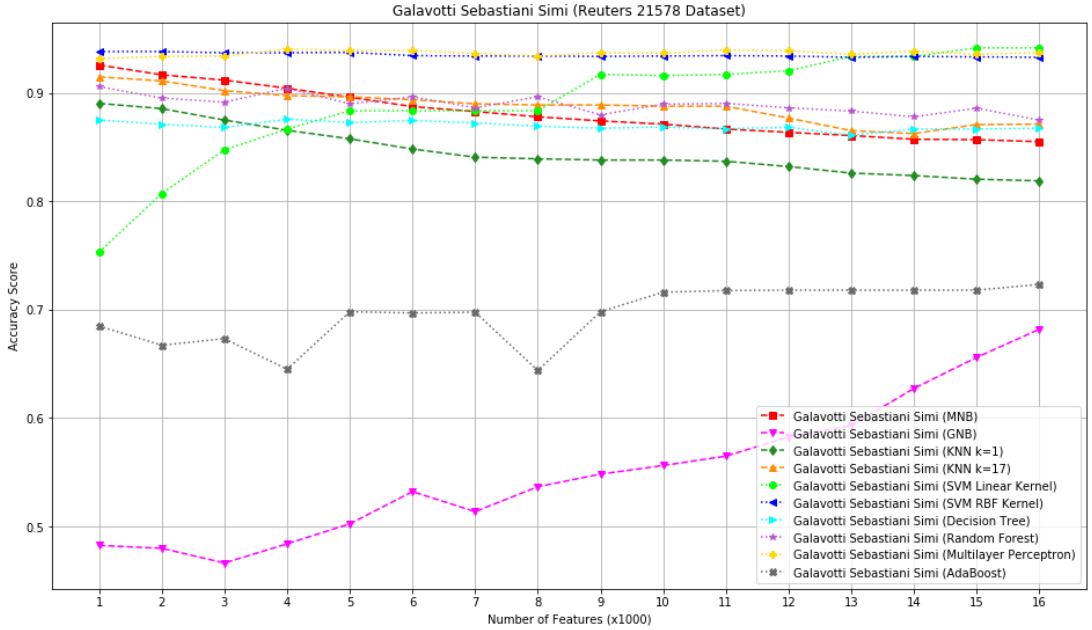

Figure 4.24: Performance of GSS on Reuters-21578

Figure 4.24 above gives information on the results of different classification algorithms when the GSS used as a feature selection method. The SVM with RBF kernel and MLP classifiers have behaved similarly, and they show the best performance while the GNB is the worst. The linear SVM improves its performance as the number of features are increases. AdaBoost works better than GNB when they compared. kNN (k=17) classifier shows the best performance after MLP and SVM (RBF kernel). All other classifiers give a similar performance.

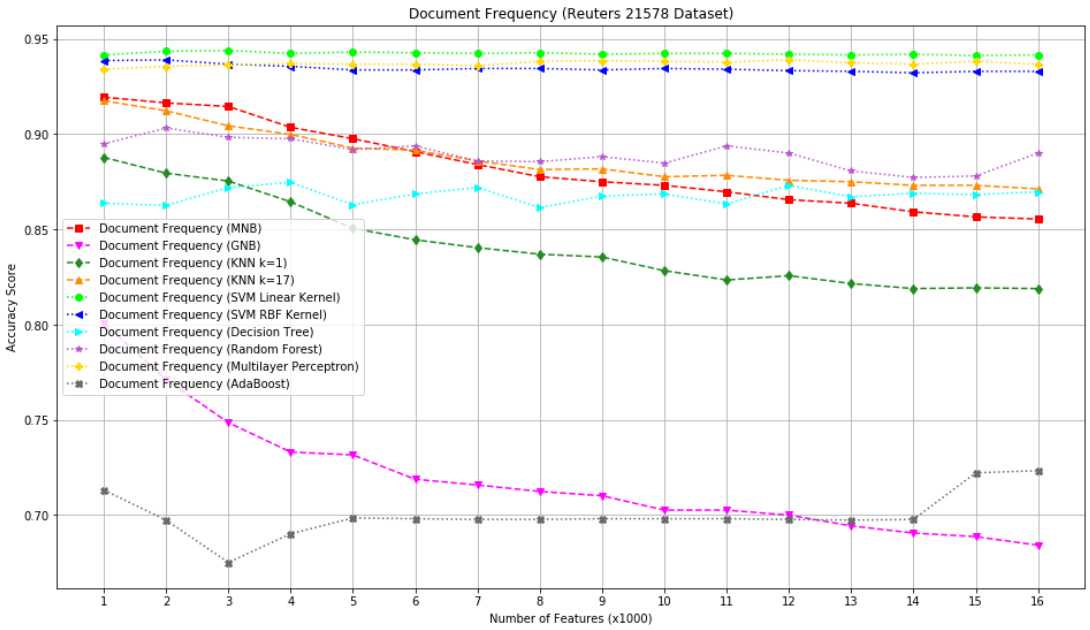

Figure 4.25: Performance of DF on Reuters-21578

Figure 4.25 provides the details on the results of different learning machines with DF feature selection method. The both SVMs and MLP classifiers show the best performance when the feature subsets selected according to DF. The AdaBoost and GNB classifiers gave the worst with DF. As seen from the figure, there is a significant decrease in the performance of classifiers as the size of the feature subset increases.

4.2.2.2 Results Based on Classification Algorithms

In this section, we provide the experimental results of assessing the performance of the classification algorithms with different feature selection metrics. The aim is to investigate and compare the differences between feature selection methods when a specific classifier used.

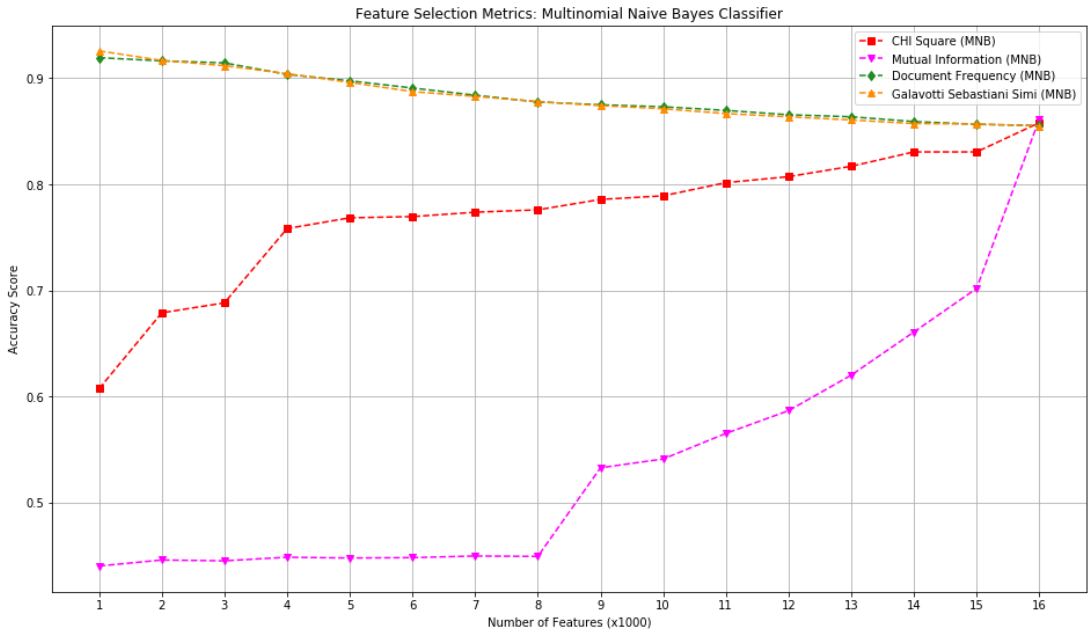

Figure 4.26: Performance of MNB on Reuters-21578

Figure 4.26 above illustrates that the performance of MNB classifier with four different feature selection metrics. As seen from the figure, MNB shows the best performance with GSS and DF. Also, the success of those two feature selection algorithms decreases as the number of features increases. The results show that the performance of MI and CHI increases linearly with the number of features.

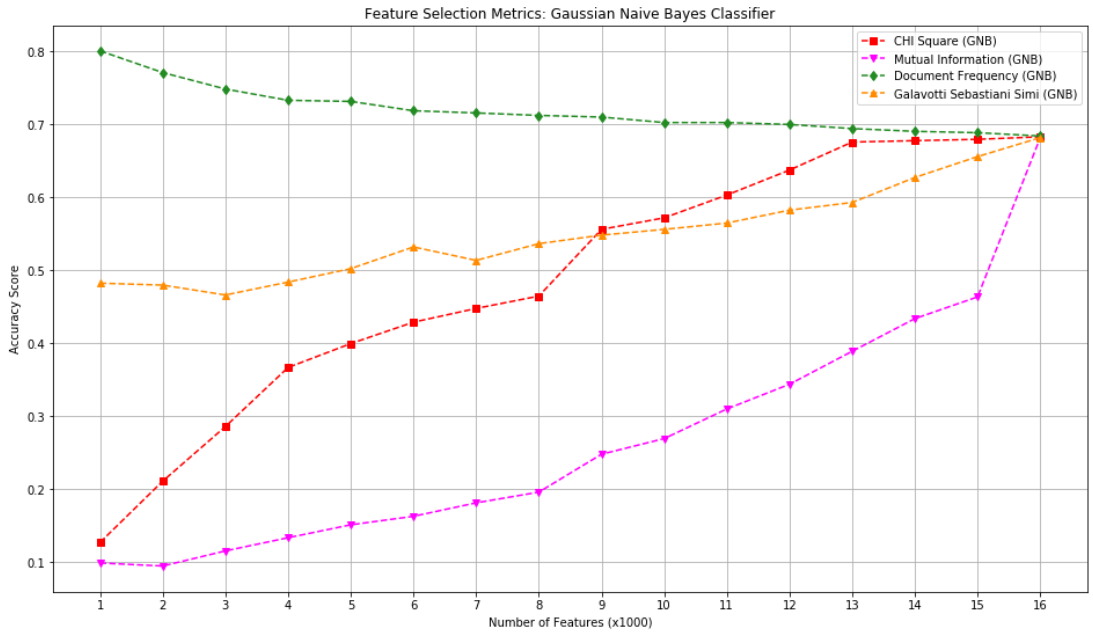

Figure 4.27: Performance of GNB on Reuters-21578

Figure 4.27 shows the results of four feature selection metrics when the GNB used as a baseline classifier. As seen from the figure, DF gives the best, and MI gives the worst performance with GNB classifier. Although the GSS and CHI show different behavior, they show the same performance when the feature subset contains 9000 distinct words.

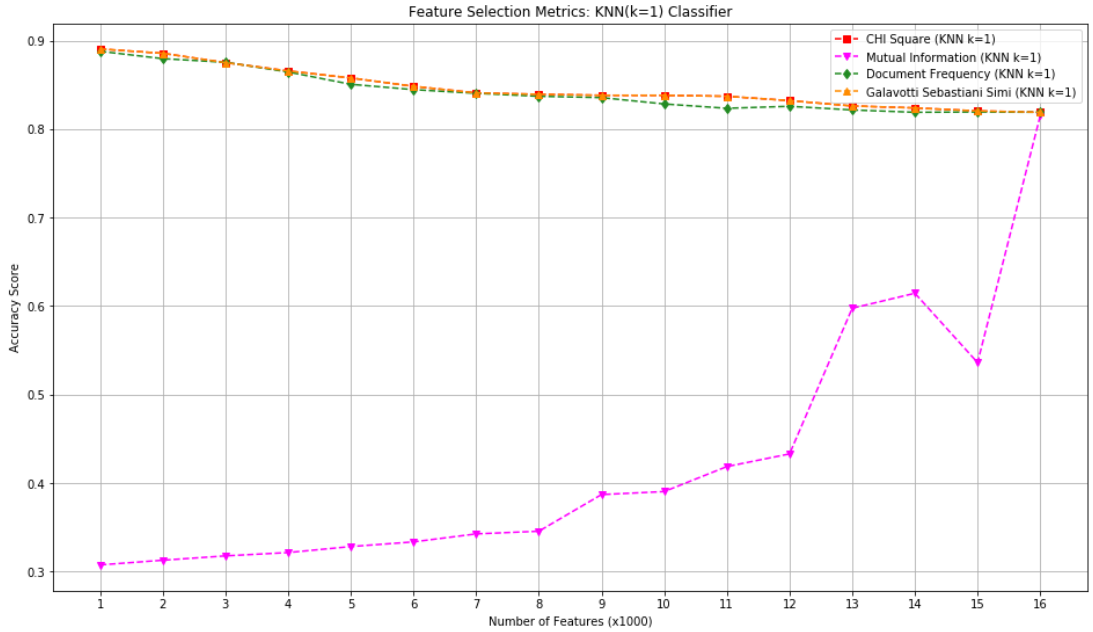

Figure 4.28: Performance of kNN(*k*=1) on Reuters-21578

Figure 4.28 provides information on the results of four different feature selection methods with kNN classifier when the *k* equals to 1. As seen from the figure, CHI, MI and DF are in the same behavior and they give the best performance with the least number of features. The MI is the worst performing feature selection metric with kNN (*k*=1).

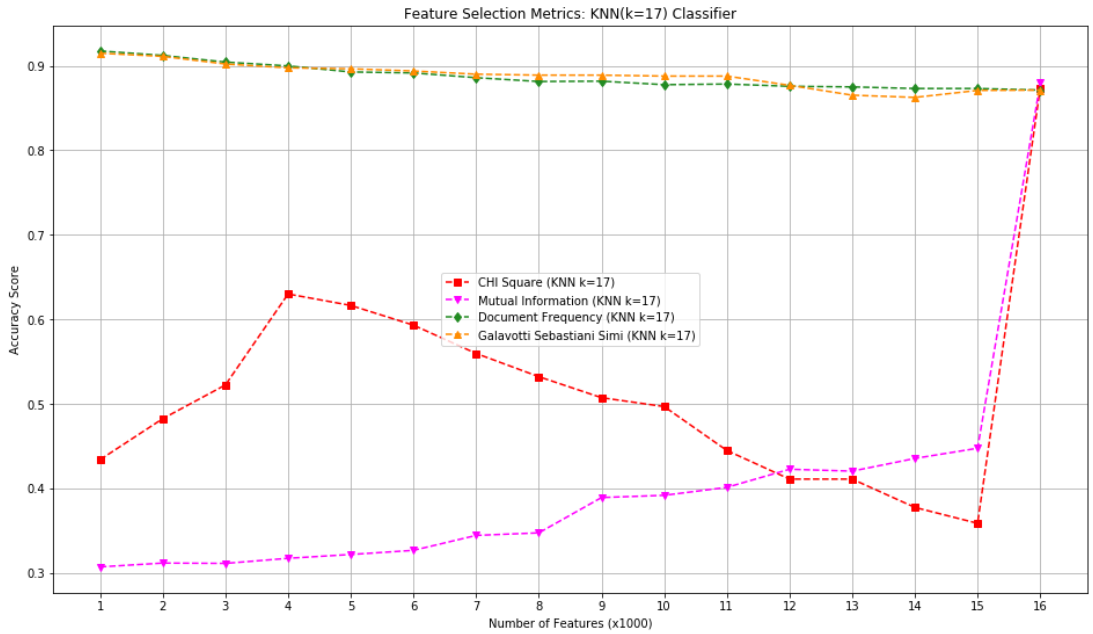

Figure 4.29: Performance of kNN (*k*=17) on Reuters-21578

Figure 4.29 above, shows the effectiveness of different feature selection algorithms with kNN classifier when the *k* parameter chose as 17. As seen from the figure, GSS and DF show the best performance when the size of the feature subset is smaller. MI gives the worst results. When the *k* is chosen as 17, the performance of CHI is decreased.

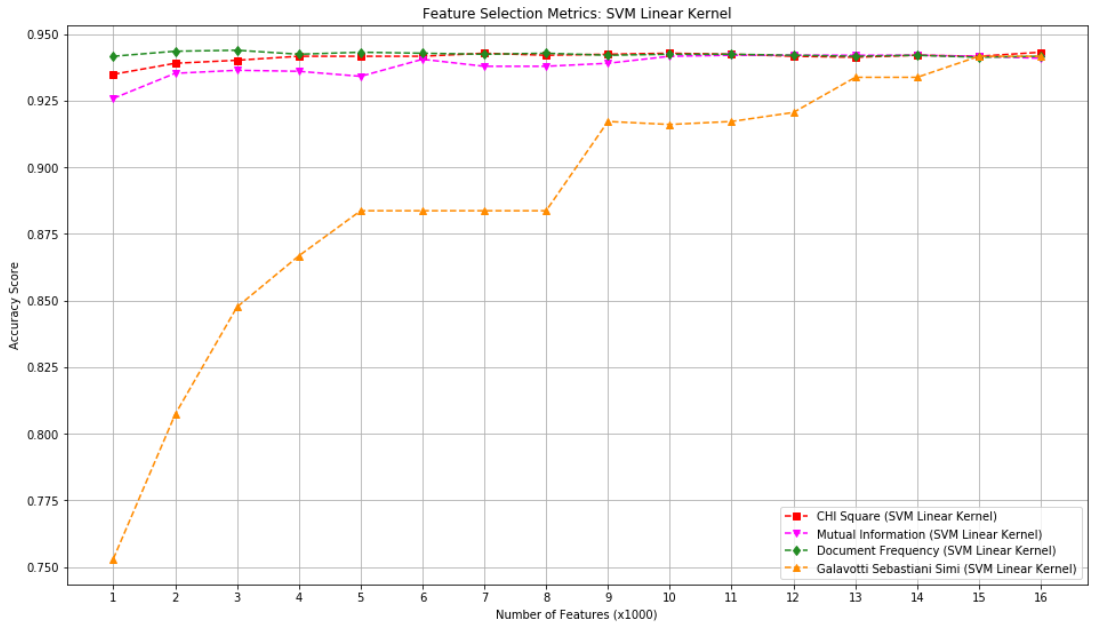

Figure 4.30: Performance of Linear SVM on Reuters-21578

As seen from the Figure 4.30, all feature selection algorithms produce good results with linear SVM except the GSS, and they show the same performance in range from 10000 to 16000 features.

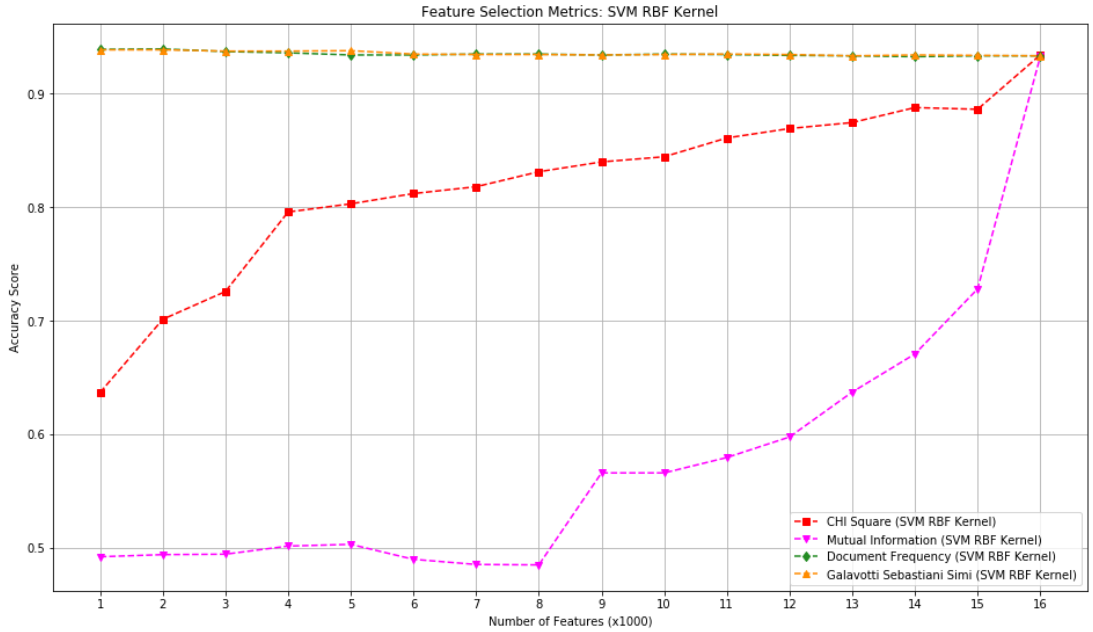

Figure 4.31: Performance of SVM (RBF Kernel) on Reuters-21578

Figure 4.31 above shows the results of four different feature selection metrics with SVM (RBF Kernel). As seen from the figure, DF and GSS show same behavior, and they give the best results. While CHI is providing the best results after DF and GSS, MI gives the worst outcomes for feature selection.

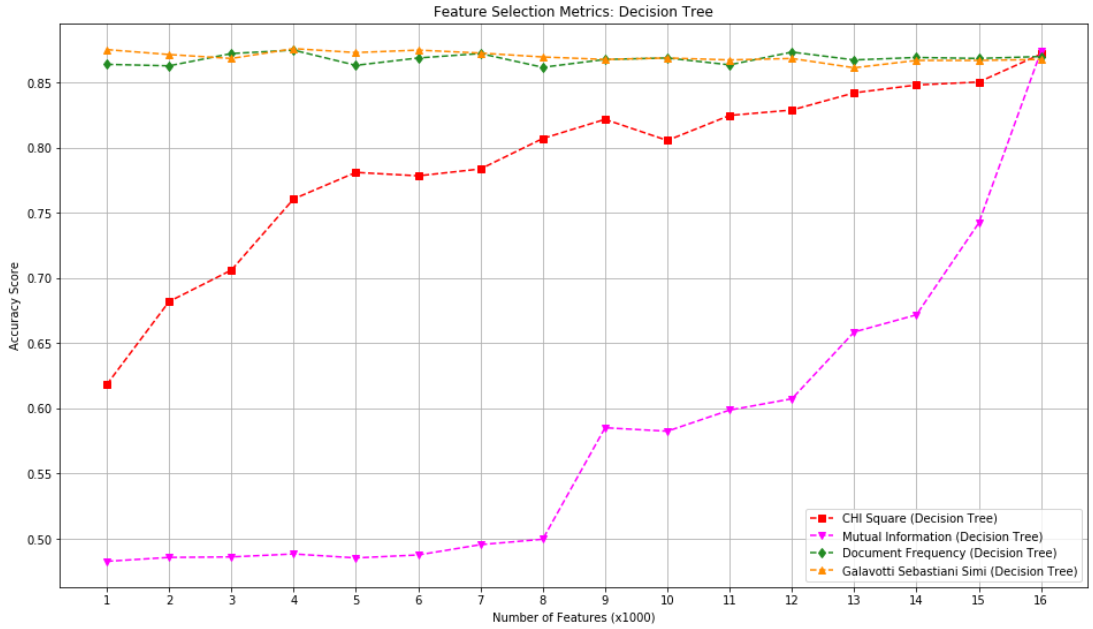

Figure 4.32: Performance of DT on Reuters-21578

Figure 4.32 shows the results of feature selection metrics with DT classifier. DF and GSS produce the best results with DT. The CHI and MI give better results with a large number of features.

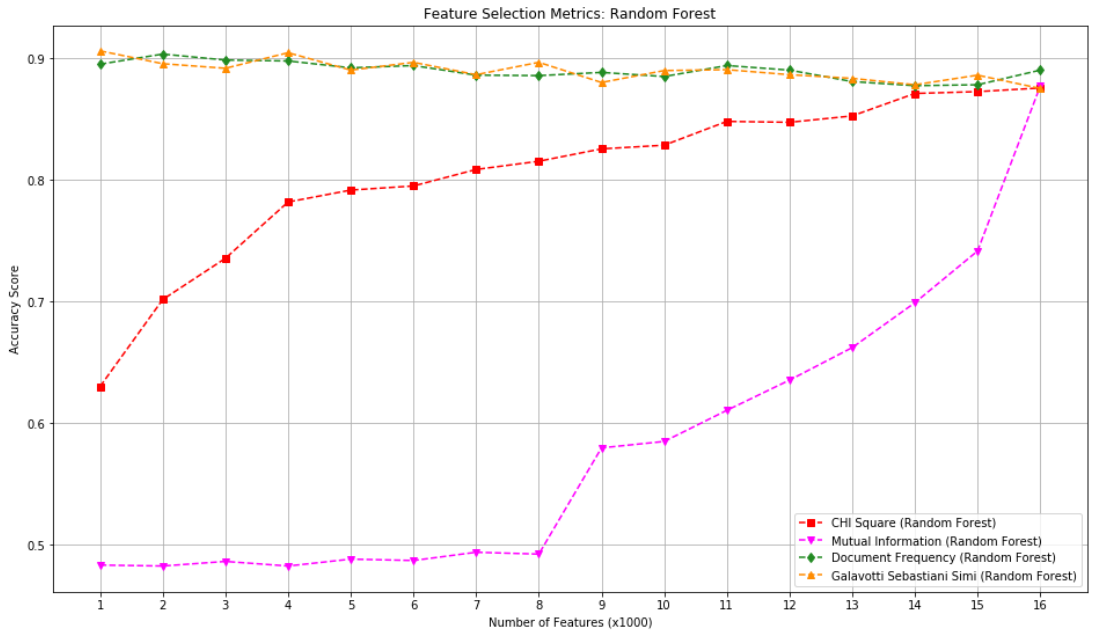

Figure 4.33: Performance of RF on Reuters-21578

Figure 4.33 above shows the performance of feature selection methods with RF classifier. As in DTs, the feature selection methods show a similar performance with RF classifier.

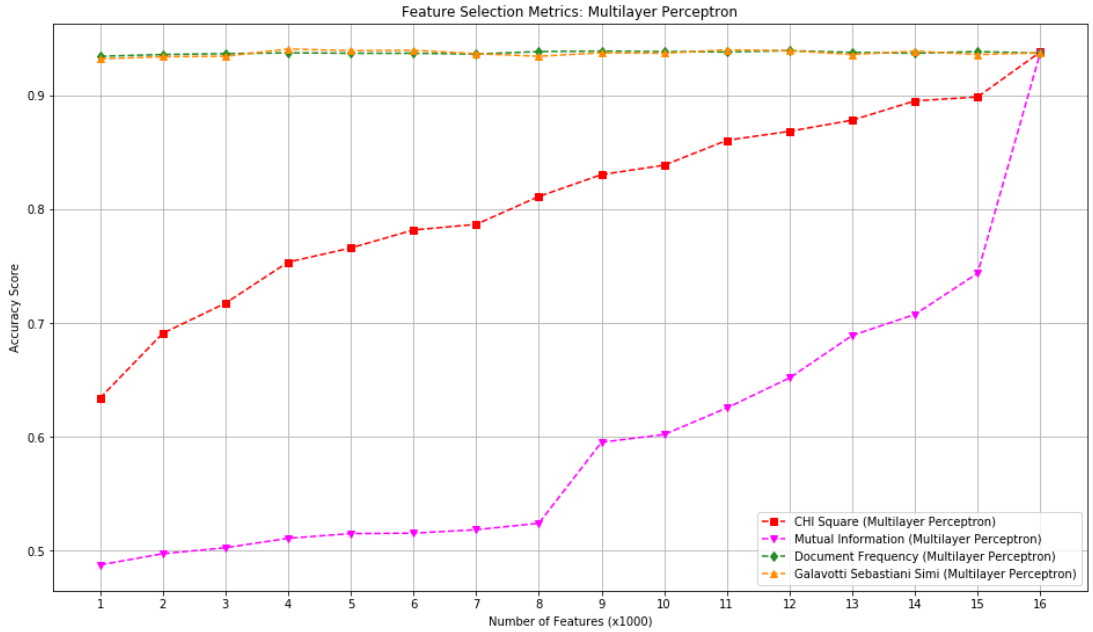

Figure 4.34: Performance of MLP on Reuters-21578

Figure 4.34, provides feature selection details with four different feature selection methods when MLP is selected for classification. As seen from the figure, the methods give similar results for feature selection as they did with SVM (RBF Kernel).

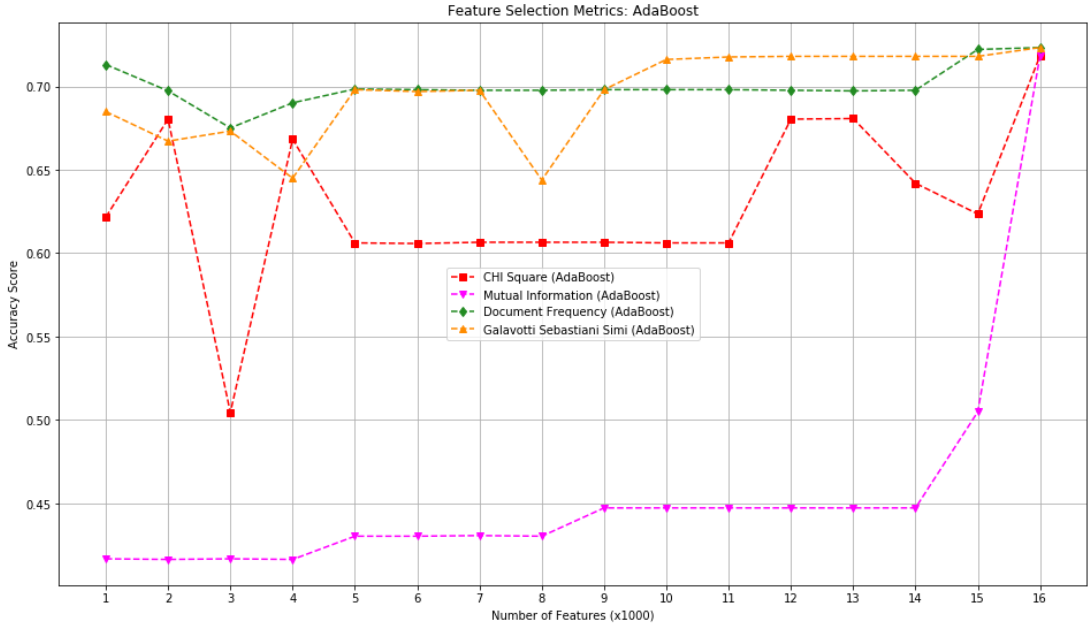

Figure 4.35: Performance of AdaBoost on Reuters-21578

Figure 4.35 above reflects the effectiveness of four different feature selection algorithms when the AdaBoost algorithm is considered for classification. As seen from the figure, DF shows the best performance with 1000 features. GSS and DF give the same results for 3000, 5000, 6000, 7000, and 9000 features. CHI shows the best result with 16000 features, and MI is the worst feature selection method for AdaBoost classifier.

4.3 Results for The 4-Universities Dataset

In this section, the results of experiments conducted with The 4-Universities dataset are presented.

4.3.1 Performance of Classification Algorithms

In this section, the performance of classification algorithms with using different term weighting schemes is given.

4.3.1.1 Using TF-IDF Weighting Scheme

The following tables provide information on the performance of classification algorithms with the 4-Universities dataset in terms of precision, recall, accuracy, *f*-measure, and micro-macro averaging metrics. The TF-IDF weighting scheme is used for weighting the terms.

Table 4.31: CR of MNB on The 4-Universities Dataset

	Multinomial Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.9681	0.8531	0.9070	320
faculty	0.7169	0.6398	0.6761	372
project	0.0000	0.0000	0.0000	175
student	0.6580	0.9788	0.7870	519
micro-avg	0.7352	0.7352	0.7352	1386
macro-avg	0.5857	0.6179	0.5925	1386
Accuracy	0.7352			

Table 4.31 shows the classification details of MNB classifier. It is seen from the table that the "project" category contains a minimum number of documents, and the MNB classifier cannot classify the documents belonging to that category. When the *f*-measure taken into consideration, 91% of the samples belonging to the "course" category, 68% of the samples belonging to the "faculty" category and 79% of the samples belonging to the "student" category correctly placed in the relevant categories. The table also shows that 74% of test documents are correctly classified. The confusion matrix of the classification can be seen from Figure 4.36 below.

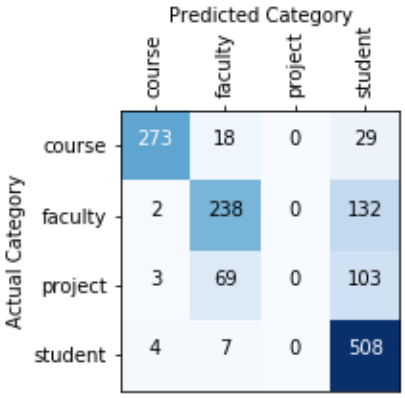

Figure 4.36: CM of MNB on The 4-Universities Dataset

Table 4.32: CR of GNB on The 4-Universities Dataset

	Gaussian Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.6985	0.7312	0.7145	320
faculty	0.5011	0.5968	0.5448	372
project	0.4113	0.2914	0.3411	175
student	0.6570	0.6127	0.6341	519
micro-avg	0.5952	0.5952	0.5952	1386
macro-avg	0.5670	0.5580	0.5586	1386
Accuracy	0.5952			

Table 4.32 reflects the performance of GNB classifier on The 4-Universities dataset. As seen from the table, according to the f-measure. GNB classifier is correctly classified 71% of documents for the “course”, 54% of documents for “faculty”, 34% of documents for “project” and 63% of documents for the “student” category. When it is compared to MNB, GNB can classify documents for the "project" category, but overall classification accuracy is 59%. The related confusion matrix can be seen from Figure 4.37 below.

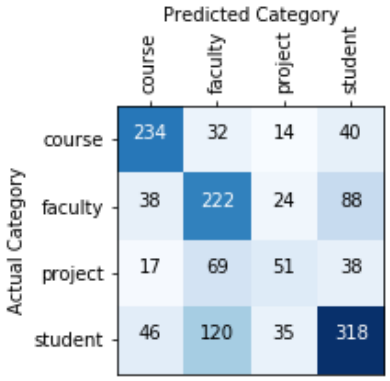

Figure 4.37: CM of GNB on The 4-Universities Dataset

Table 4.33: CR of kNN (k=1) on The 4-Universities Dataset

	kNN (*k*=1) Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.4825	0.3875	0.4298	320
faculty	0.8871	0.1478	0.2535	372
project	0.1522	0.1200	0.1342	175
student	0.4349	0.7784	0.5580	519
micro-avg	0.4358	0.4358	0.4358	1386
macro-avg	0.4892	0.3584	0.3439	1386
Accuracy	0.4358			

Table 4.33 shows the classification result of kNN (*k*=1) classifier. It is seen from the table that, since the "student" category contains the most examples and the *k* parameter is selected as 1, kNN has classified most of the documents belonging to this class with a success of 56%. kNN is classified as 44% of the documents correctly. The confusion matrix of classification process can be seen from Figure 4.38 below.

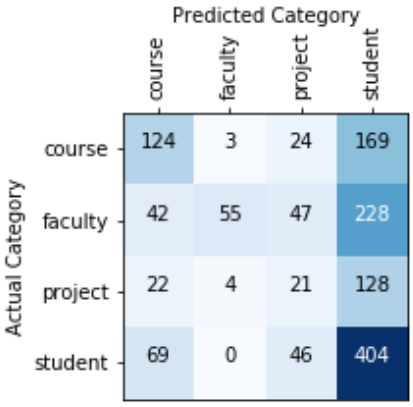

Figure 4.38: CM of kNN (*k*=1) on The 4-Universities Dataset

Table 4.34: CR of kNN (k=17) on The 4-Universities Dataset

	kNN (*k*=17) Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.9286	0.7312	0.8182	320
faculty	0.9605	0.1962	0.3259	372
project	0.5000	0.0229	0.0437	175
student	0.4810	0.9730	0.6437	519
micro-avg	0.5887	0.5887	0.5887	1386
macro-avg	0.7175	0.4808	0.4579	1386
Accuracy	0.5887			

Table 4.34 gives the details of the classification process for kNN (*k*=17) classifier. When it is compared with Table 4.45, the choosing of larger *k* value, the classification accuracy is improved from 44% to 59%. According to the *f*-measure, it is seen that the performance of the classifier is better than kNN (*k*=1) except the “project” category. The relevant confusion matrix can be seen from Figure 4.39.

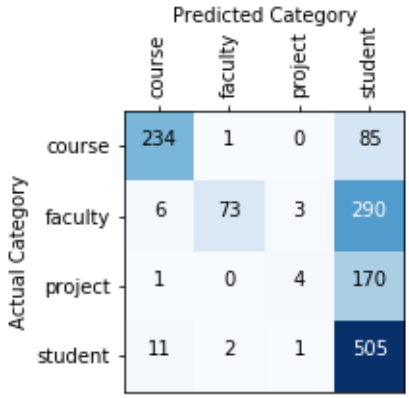

Figure 4.39: CM of kNN (*k*=17) on The 4-Universities Dataset

Table 4.35: CR of SVM (Linear Kernel) on The 4-Universities Dataset

	SVM (Linear Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.9467	0.9437	0.9452	320
faculty	0.8529	0.8575	0.8552	372
project	0.9149	0.7371	0.8165	175
student	0.8732	0.9287	0.9001	519
micro-avg	0.8889	0.8889	0.8889	1386
macro-avg	0.8969	0.8668	0.8793	1386
Accuracy	0.8889			

Table 4.35 shows the performance of linear SVM for The 4-Universities dataset. It is seen from the table that linear SVM gives the considerable performance for all categories according to the *f*-measure and it shows the best performance for the "project" class which contains the minimum number of samples. The confusion matrix of classification can be seen from Figure 4.40 below.

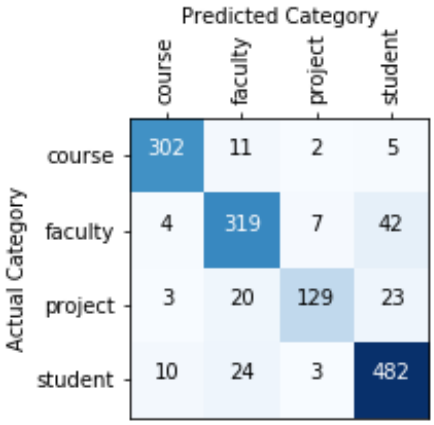

Figure 4.40: CM of linear SVM on The 4-Universities Dataset

Table 4.36: CR of SVM (RBF Kernel) on The 4-Universities Dataset

	SVM (RBF Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.0000	0.0000	0.0000	320
faculty	0.0000	0.0000	0.0000	372
project	0.0000	0.0000	0.0000	175
student	0.3745	1.0000	0.5449	519
micro-avg	0.3745	0.3745	0.3745	1386
macro-avg	0.0936	0.2500	0.1362	1386
Accuracy	0.3745			

Table 4.36 depicts the classification performance of SVM (RBF Kernel). Since we used default parameter settings, RBF Kernel cannot classify the samples for “course”, “faculty”, and “project” classes. The confusion matrix for the classification process can be seen from Figure 4.41 below.

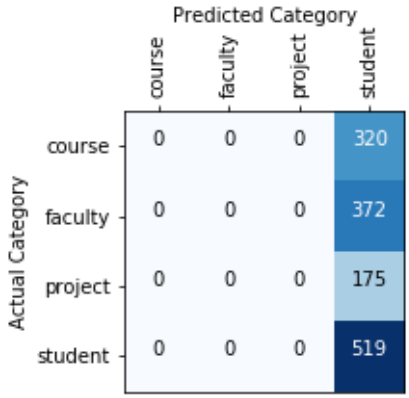

Figure 4.41: CM of SVM (RBF Kernel) on The 4-Universities Dataset

Table 4.37: CR of DT on The 4-Universities Dataset

	Decision Tree Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.7879	0.8125	0.8000	320
faculty	0.7456	0.6774	0.7099	372
project	0.5833	0.5200	0.5498	175
student	0.7313	0.7919	0.7604	519
micro-avg	0.7316	0.7316	0.7316	1386
macro-avg	0.7120	0.7005	0.7050	1386
Accuracy	0.7316			

Table 4.37 above shows that the performance of DT classifier. When it is compared with other classifiers, the performance of DT is close to MNB classifier, and it is better than MNB for placing the documents to "project" class. It is seen from the table that DT is correctly classified the 73% of documents. The confusion matrix can be seen from Figure 4.42 below.

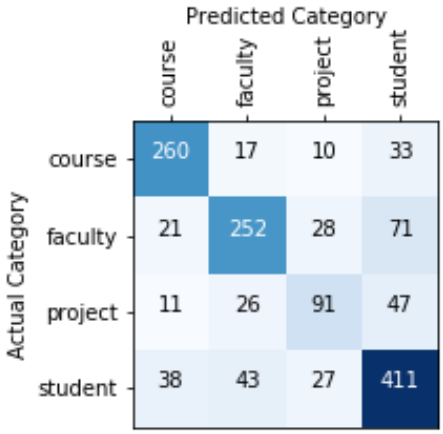

Figure 4.42: CM of DT on The 4-Universities Dataset

Table 4.38: CR of RF on The 4-Universities Dataset

	Random Forest Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.8480	0.8719	0.8598	320
faculty	0.7031	0.7258	0.7143	372
project	0.7879	0.2971	0.4315	175
student	0.7397	0.8651	0.7975	519
micro-avg	0.7576	0.7576	0.7576	1386
macro-avg	0.7697	0.6900	0.7008	1386
Accuracy	0.7576			

Table 4.38 reflects the performance of RF. When the *f*-measure is considered, it is seen from the table that the performance of RF is better than DT except for the "project" class. It also shows a better performance than DT according to the accuracy score. The confusion matrix can be seen from Figure 4.43 below.

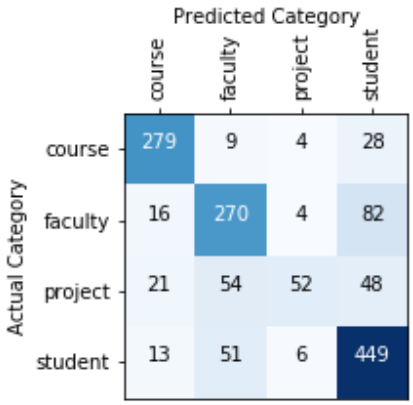

Figure 4.43: CM of RF on The 4-Universities Dataset

Table 4.39: CR of MLP on The 4-Universities Dataset

	Multilayer Perceptron (MLP) Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.9506	0.9625	0.9565	320
faculty	0.8319	0.7984	0.8148	372
project	0.8322	0.6800	0.7484	175
student	0.8452	0.9152	0.8788	519
micro-avg	0.8651	0.8651	0.8651	1386
macro-avg	0.8650	0.8390	0.8496	1386
Accuracy	0.8651			

Table 4.39 above gives the classification details for MLP classifier. As seen from the table, MLP shows the considerable performance on categorization for all classes and it is the best classifier after the linear SVM. The confusion matrix can be seen from Figure 4.44 below.

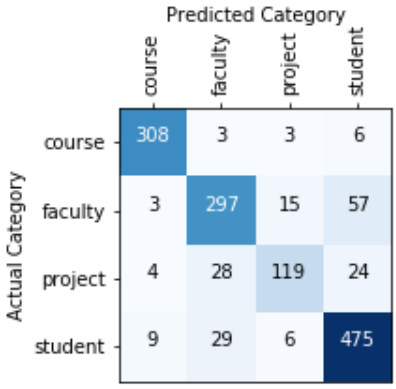

Figure 4.44: CM of MLP on The 4-Universities Dataset

Table 4.40: CR of AdaBoost on The 4-Universities Dataset

	AdaBoost Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.9320	0.8562	0.8925	320
faculty	0.7906	0.5887	0.6749	372
project	0.6503	0.6800	0.6648	175
student	0.7152	0.8709	0.7854	519
micro-avg	0.7677	0.7677	0.7677	1386
macro-avg	0.7720	0.7490	0.7544	1386
Accuracy	0.7677			

Table 4.40 shows the classification details of AdaBoost classifier. According to the *f*-measure, AdaBoost is correctly classified 89% of documents for “course”, 67% of documents for “faculty”, 66% of documents for “project” and 79% of documents for “student” category. It performed the classification process with 77% of success in overall point of view. The confusion matrix can be seen from Figure 4.45 below.

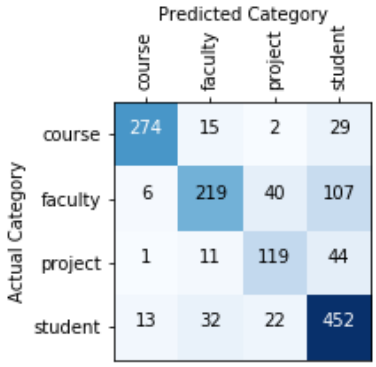

Figure 4.45: CM of AdaBoost on The 4-Universities Dataset

4.3.1.2 Using DF Weighting Scheme

The following tables give detailed information on the performance of classification algorithms with The 4-Universities dataset in terms of precision, recall, *f*-measure, and micro-macro averaging metrics. The DF weighting scheme is used for weighting the terms.

Table 4.41: CR of MNB with DF Weighting

	Multinomial Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.8809	0.8781	0.8795	320
faculty	0.7300	0.7124	0.7211	372
project	0.7200	0.7200	0.7200	175
student	0.7543	0.7688	0.7615	519
micro-avg	0.7727	0.7727	0.7727	1386
macro-avg	0.7713	0.7698	0.7705	1386
Accuracy	0.7727			

Table 4.41 shows the classification performance of MNB when the DF used as a term weighting scheme. When it is compared with TF-IDF weighting, DF shows better performance for categorizing the documents of infrequent classes and it improves the classification accuracy for MNB classifier.

Table 4.42: CR of SVM (Linear Kernel) with DF Weighting

	SVM (Linear Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.7977	0.8500	0.8230	320
faculty	0.6759	0.7177	0.6962	372
project	0.6709	0.6057	0.6366	175
student	0.8130	0.7707	0.7913	519
micro-avg	0.7540	0.7540	0.7540	1386
macro-avg	0.7394	0.7360	0.7368	1386
Accuracy	0.7540			

Table 4.42 shows that using the DF weighting scheme instead of TF-IDF negatively affects the performance of linear SVM by reducing the classification accuracy from 88% to 75%.

4.3.1.3 Using Probability-based Term Weighting Approach

The tables below reflect the effects of using probability-based term weighting scheme.

Table 4.43: CR of MNB with Probability-based Term Weighting

	Multinomial Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.4178	0.3812	0.3987	320
faculty	0.4291	0.3333	0.3752	372
project	0.1644	0.4800	0.2449	175
student	0.5816	0.3295	0.4207	519
micro-avg	0.3615	0.3615	0.3615	1386
macro-avg	0.3982	0.3810	0.3599	1386
Accuracy	0.3615			

Table 4.43 shows that the effects of applying the probability-based term weighting approach to classification task with MNB.

Table 4.44: CR of SVM (Linear Kernel) with Probability-based Term Weighting

	SVM (Linear Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
course	0.5878	0.4500	0.5097	320
faculty	0.6271	0.1989	0.3020	372
project	0.7333	0.0629	0.1158	175
student	0.4365	0.8478	0.5763	519
micro-avg	0.4827	0.4827	0.4827	1386
macro-avg	0.5962	0.3899	0.3760	1386
Accuracy	0.4827			

Table 4.44 shows the classification result of SVM (Linear Kernel) when the probability-based approach is used as a term weighting method.

As can be seen from the tables above, SVM (Linear Kernel) shows the best performance when using TF-IDF weighting method for The 4-Universities dataset.

When DF weighting used in our experiments conducted with The 4-Universities dataset, the performance is better than TF-IDF, but the performance of SVM is decreased.

As a result of the experiments performed by using probability-based term weighting method, a decrease is observed on the performance of classification algorithms.

4.3.2 Performance of Feature Selection Methods

In this section, we provide the experimental results of assessing the performance of the feature selection metrics with two different classification algorithms. The aim is to investigate the effectiveness of feature selection metrics and their achievements with different classifiers.

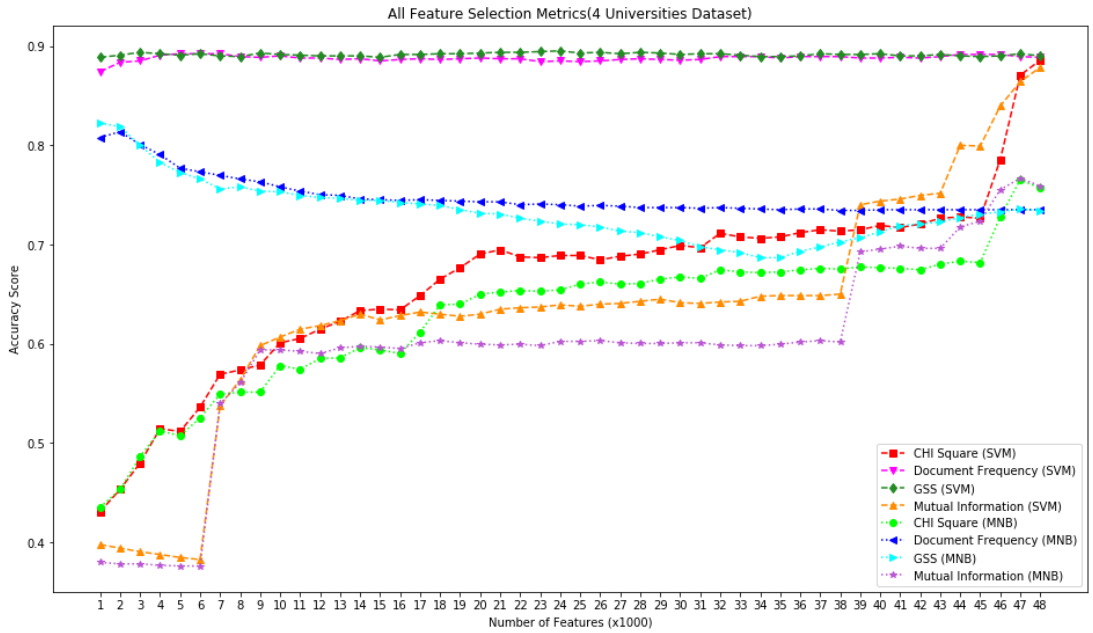

Figure 4.46: Performance of Feature Selection Metrics on The 4-Universities Dataset

Figure 4.46 reflects the performance of the four feature selection metrics with two different classifiers, MNB and SVM. As seen from the figure, GSS and DF show the best performance for The 4-Universities dataset. The CHI and MI exhibit the same behavior up to the 7000 features and show higher performance, especially after 38000 features, as the number of features selected increases.

4.4 Results for 20 Newsgroups Dataset

In this section, the results of experiments conducted with 20-Newsgroups dataset are presented.

4.4.1 Performance of Classification Algorithms

In this section, the performance of classification algorithms with using different term weighting schemes is given.

4.4.1.1 Using TF-IDF Weighting Scheme

The following tables provide information on the performance of different classification algorithms with the 20 Newsgroups dataset in terms of precision, recall,

accuracy, *f*-measure, and micro-macro averaging metrics. The TF-IDF weighting scheme is used for weighting the terms.

Table 4.45: CR of MNB on 20 Newsgroups Dataset

	Multinomial Navie Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.8982	0.8731	0.8855	394
comp.windows.x	0.9091	0.8608	0.8843	395
rec.autos	0.9593	0.9520	0.9556	396
rec.motorcycles	0.9578	0.9698	0.9638	398
rec.sport.baseball	0.9792	0.9496	0.9642	397
rec.sport.hockey	0.9517	0.9875	0.9692	399
sci.crypt	0.8813	0.9747	0.9257	396
sci.med	0.9801	0.8687	0.9210	396
sci.space	0.9446	0.9518	0.9482	394
soc.religion.christian	0.9176	0.9799	0.9478	398
micro-avg	0.9369	0.9369	0.9369	3963
macro-avg	0.9379	0.9368	0.9365	3963
Accuracy	0.9369			

Table 4.45 gives information on classification performance of MNB classifier. As seen from the table, the dataset has balanced document distribution and 94% of test documents correctly categorized by MNB. According to *f*-measure MNB performed best for category “rec.sport.hockey” and worst for the category “comp.os.ms-windows.misc”. The confusion matrix can be seen from Figure 4.47 below.

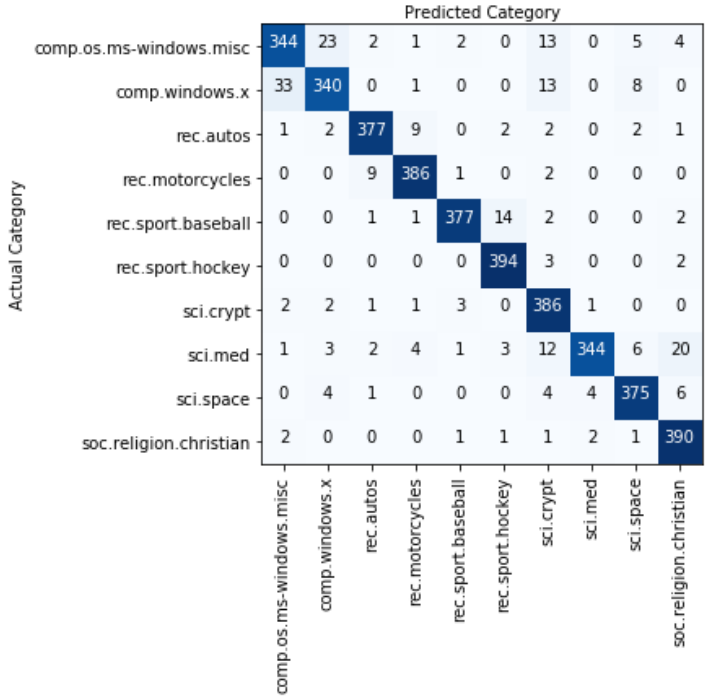

Figure 4.47: CM of MNB on 20 Newsgroups Dataset

Table 4.46: CR of GNB on 20 Newsgroups Dataset

	Gaussian Navie Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.8055	0.5990	0.6870	394
comp.windows.x	0.8123	0.7671	0.7891	395
rec.autos	0.8519	0.8131	0.8320	396
rec.motorcycles	0.7712	0.8467	0.8072	398
rec.sport.baseball	0.9081	0.8212	0.8624	397
rec.sport.hockey	0.9077	0.9123	0.9100	399
sci.crypt	0.7834	0.9040	0.8394	396
sci.med	0.7482	0.7955	0.7711	396
sci.space	0.7432	0.8299	0.7842	394
soc.religion.christian	0.9035	0.9171	0.9102	398
micro-avg	0.8208	0.8208	0.8208	3963
macro-avg	0.8235	0.8206	0.8193	3963
Accuracy	0.8208			

Table 4.46 shows the classification details of GNB classifier. As seen from the table, it has lower performance rate than MNB, and 82% of test documents were correctly categorized by GNB. The confusion matrix can be seen from Figure 4.48 below.

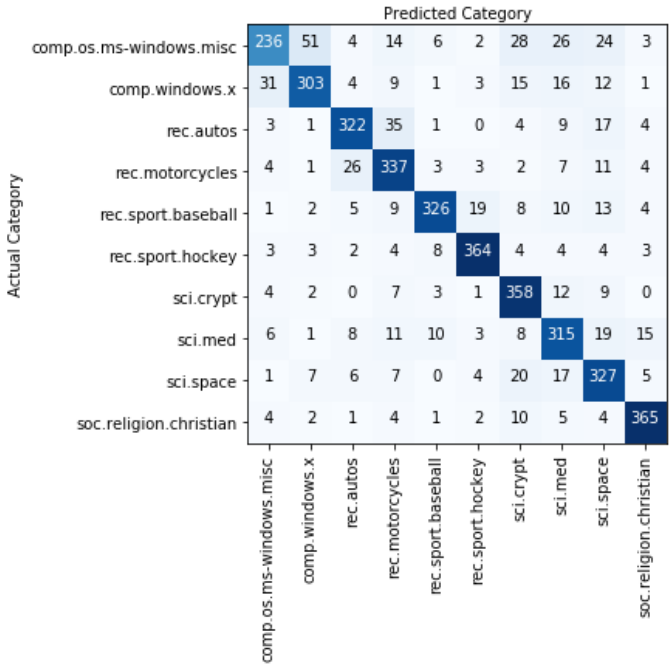

Figure 4.48: CM of GNB on 20 Newsgroups Dataset

Table 4.47: CR of kNN (k=1) on 20 Newsgroups Dataset

	kNN (*k*=1) Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.7720	0.7563	0.7641	394
comp.windows.x	0.7749	0.7671	0.7710	395
rec.autos	0.8372	0.8308	0.8340	396
rec.motorcycles	0.8501	0.8693	0.8596	398
rec.sport.baseball	0.8589	0.8589	0.8589	397
rec.sport.hockey	0.8256	0.9373	0.8779	399
sci.crypt	0.8997	0.8838	0.8917	396
sci.med	0.8626	0.7929	0.8263	396
sci.space	0.8802	0.8579	0.8689	394
soc.religion.christian	0.9223	0.9246	0.9235	398
micro-avg	0.8481	0.8481	0.8481	3963
macro-avg	0.8484	0.8479	0.8476	3963
Accuracy	0.8481			

Table 4.47 shows the classification details of kNN (*k*=1). According to the *f*-measure, 85% of test documents correctly clasisified into the corresponding category. The confusion matrix can be seen from Figure 4.49 below.

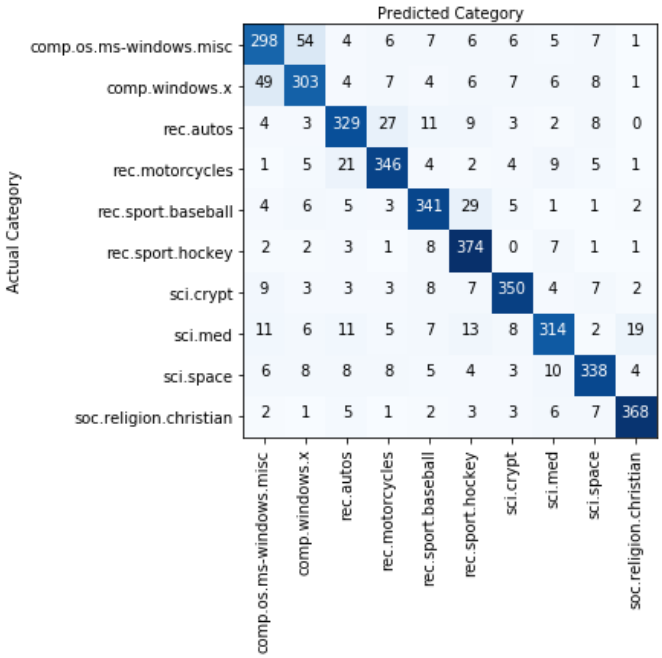

Figure 4.49: CM of kNN (*k*=1) on 20 Newsgroups Dataset

Table 4.48: CR of kNN (k=17) on 20 Newsgroups Dataset

	kNN (*k*=17) Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.8146	0.8807	0.8463	394
comp.windows.x	0.8791	0.8101	0.8432	395
rec.autos	0.8442	0.9167	0.8789	396
rec.motorcycles	0.9213	0.8819	0.9012	398
rec.sport.baseball	0.9199	0.8967	0.9082	397
rec.sport.hockey	0.8562	0.9850	0.9161	399
sci.crypt	0.9545	0.9545	0.9545	396
sci.med	0.9502	0.7702	0.8508	396
sci.space	0.9418	0.9036	0.9223	394
soc.religion.christian	0.9002	0.9523	0.9255	398
micro-avg	0.8953	0.8953	0.8953	3963
macro-avg	0.8982	0.8952	0.8947	3963
Accuracy	0.8953			

Table 4.48 shows the classification details of kNN (*k*=17). As seen from the table, choosing of larger *k* value is improved the *f*-measure scores and classification accuracy from 85% to 90%. The confusion matrix can be seen from Figure 4.50 below.

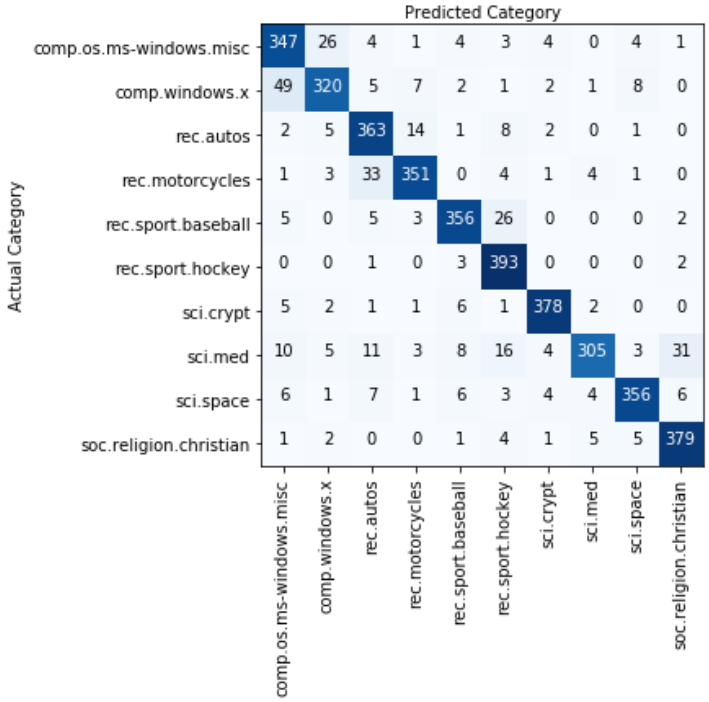

Figure 4.50: CM of kNN (*k*=17) on 20 Newsgroups Dataset

Table 4.49: CR of SVM (Linear Kernel) on 20 Newsgroups Dataset

	SVM (Linear Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.8753	0.8909	0.8830	394
comp.windows.x	0.8589	0.8633	0.8611	395
rec.autos	0.9203	0.9621	0.9407	396
rec.motorcycles	0.9616	0.9447	0.9531	398
rec.sport.baseball	0.9387	0.9647	0.9516	397
rec.sport.hockey	0.9696	0.9599	0.9647	399
sci.crypt	0.9919	0.9242	0.9569	396
sci.med	0.9109	0.9293	0.9200	396
sci.space	0.9585	0.9391	0.9487	394
soc.religion.christian	0.9548	0.9548	0.9548	398
micro-avg	0.9334	0.9334	0.9334	3963
macro-avg	0.9341	0.9333	0.9335	3963
Accuracy	0.9334			

Table 4.49 gives the classification details of linear SVM. As seen from the table, SVM has considerable performance. The success rate of SVM on the classification process is 93%. The confusion matrix of linear SVM can be seen from Figure 4.51 below.

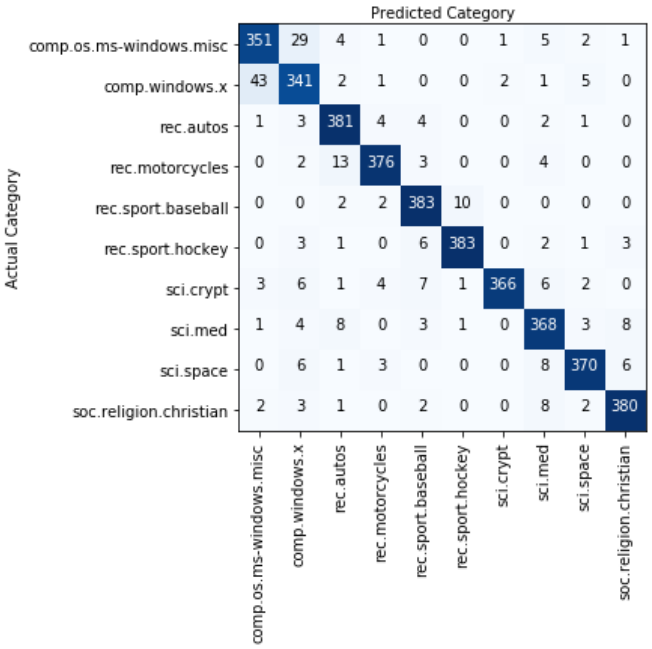

Figure 4.51: CM of SVM (Linear Kernel) on 20 Newsgroups Dataset

Table 4.50: CR of SVM (RBF Kernel) on 20 Newsgroups Dataset

	SVM (RBF Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.0000	0.0000	0.0000	394
comp.windows.x	0.0000	0.0000	0.0000	395
rec.autos	0.0000	0.0000	0.0000	396
rec.motorcycles	0.0000	0.0000	0.0000	398
rec.sport.baseball	0.0000	0.0000	0.0000	397
rec.sport.hockey	0.1007	1.0000	0.1829	399
sci.crypt	0.0000	0.0000	0.0000	396
sci.med	0.0000	0.0000	0.0000	396
sci.space	0.0000	0.0000	0.0000	394
soc.religion.christian	0.0000	0.0000	0.0000	398
micro-avg	0.1007	0.1007	0.1007	3963
macro-avg	0.0101	0.1000	0.0183	3963
Accuracy	0.1007			

Table 4.50 above shows the classification report of SVM with RBF kernel. As seen from the table, SVM with RBF kernel is insufficient to classify the test documents because it used with default parameter settings, and it shows the worst performance with 10% of accuracy score because we did not optimize the parameters for 20 Newsgroups dataset. The confusion matrix related to such classification can be seen from Figure 4.52 below.

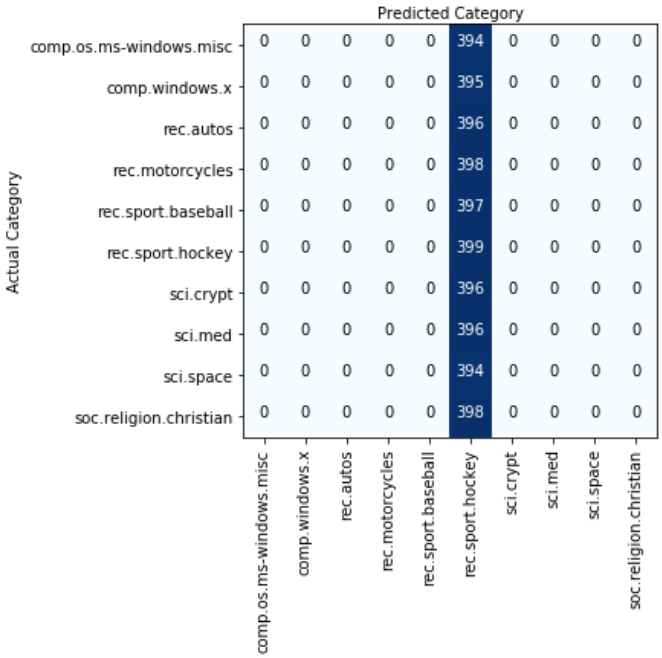

Figure 4.52: CM of SVM (RBF Kernel) on 20 Newsgroups Dataset

Table 4.51: CR of DT on 20 Newsgroups Dataset

	Decision Tree Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.6490	0.6853	0.6667	394
comp.windows.x	0.6632	0.6380	0.6503	395
rec.autos	0.6969	0.7374	0.7166	396
rec.motorcycles	0.8350	0.8266	0.8308	398
rec.sport.baseball	0.6833	0.7607	0.7199	397
rec.sport.hockey	0.8224	0.7544	0.7869	399
sci.crypt	0.8333	0.7955	0.8140	396
sci.med	0.6097	0.6667	0.6369	396
sci.space	0.7806	0.7132	0.7454	394
soc.religion.christian	0.8747	0.8241	0.8486	398
micro-avg	0.7403	0.7403	0.7403	3963
macro-avg	0.7448	0.7402	0.7416	3963
Accuracy	0.7403			

Table 4.51 provides information on classification details of DT classifier. As seen from the table, 74% of test documents correctly placed into the corresponding category. The confusion matrix of classification can be seen from Figure 4.53 below.

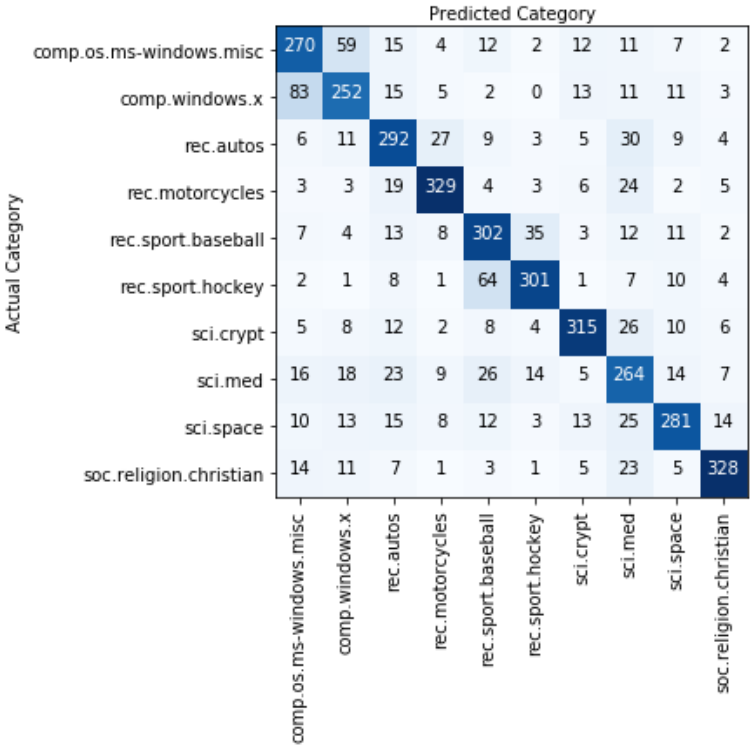

Figure 4.53: CM of DT on 20 Newsgroups Dataset

Table 4.52: CR of RF on 20 Newsgroups Dataset

	Random Forest Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.5760	0.8655	0.6917	394
comp.windows.x	0.6578	0.6278	0.6425	395
rec.autos	0.6982	0.7828	0.7381	396
rec.motorcycles	0.8613	0.8266	0.8436	398
rec.sport.baseball	0.7518	0.8086	0.7791	397
rec.sport.hockey	0.8760	0.8321	0.8535	399
sci.crypt	0.9493	0.8030	0.8700	396
sci.med	0.8377	0.6515	0.7330	396
sci.space	0.8928	0.7817	0.8336	394
soc.religion.christian	0.9519	0.8945	0.9223	398
micro-avg	0.7875	0.7875	0.7875	3963
macro-avg	0.8053	0.7874	0.7907	3963
Accuracy	0.7875			

Table 4.52 reflects the classification details of RF classifier. As seen from the table, RF shows better performance than DT according to the classification accuracy and *f*-measure scores. The confusion matrix can be seen from Figure 4.54 below.

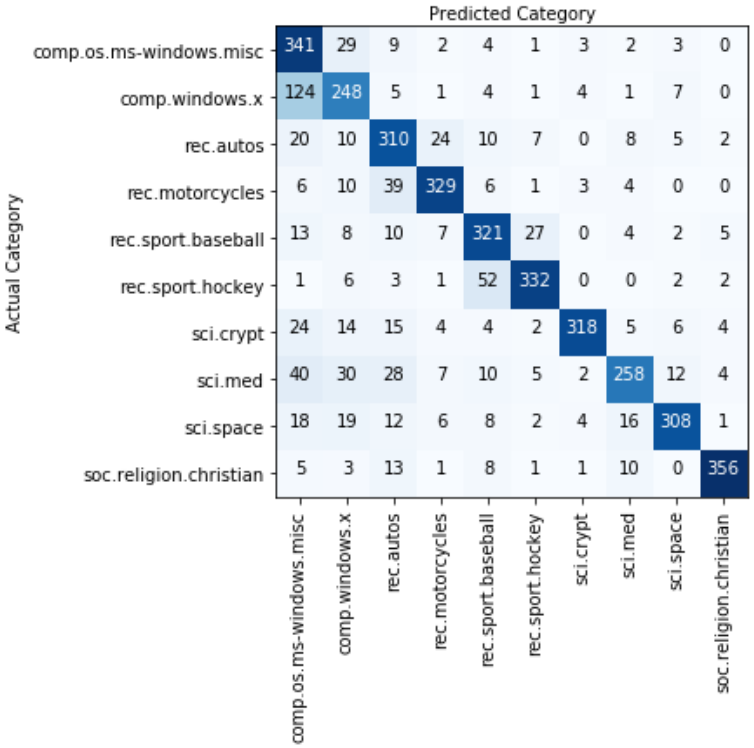

Figure 4.54: CM of RF on 20 Newsgroups Dataset

Table 4.53: CR of MLP on 20 Newsgroups Dataset

	Multilayer Perceptron (MLP) Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.8949	0.8858	0.8903	394
comp.windows.x	0.8803	0.8937	0.8869	395
rec.autos	0.9619	0.9571	0.9595	396
rec.motorcycles	0.9580	0.9749	0.9664	398
rec.sport.baseball	0.9556	0.9748	0.9651	397
rec.sport.hockey	0.9681	0.9875	0.9777	399
sci.crypt	0.9869	0.9545	0.9705	396
sci.med	0.9578	0.9167	0.9368	396
sci.space	0.9640	0.9518	0.9579	394
soc.religion.christian	0.9439	0.9724	0.9579	398
micro-avg	0.9470	0.9470	0.9470	3963
macro-avg	0.9471	0.9469	0.9469	3963
Accuracy	0.9470			

Table 4.53 above shows the classification details of MLP classifier. MLP is the classifier that gives the best results for 20 Newsgroups dataset, and it outperformed the linear SVM. As seen from the table, 95% of test documents are correctly categorized. The related confusion matrix can be seen from Figure 4.55 below.

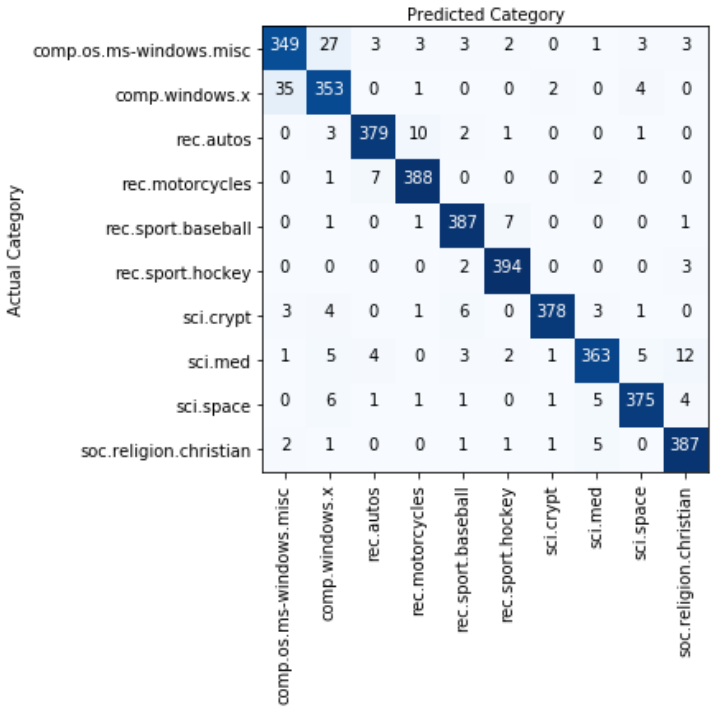

Figure 4.55: CM of MLP on 20 Newsgroups Dataset

Table 4.54: CR of AdaBoost on 20 Newsgroups Dataset

	AdaBoost Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.6936	0.6091	0.6486	394
comp.windows.x	0.6085	0.3620	0.4540	395
rec.autos	0.7972	0.5758	0.6686	396
rec.motorcycles	0.9194	0.7739	0.8404	398
rec.sport.baseball	0.7424	0.6171	0.6740	397
rec.sport.hockey	0.8128	0.7619	0.7865	399
sci.crypt	0.9391	0.8182	0.8745	396
sci.med	0.3441	0.8636	0.4921	396
sci.space	0.8107	0.6954	0.7486	394
soc.religion.christian	0.8921	0.8518	0.8715	398
micro-avg	0.6932	0.6932	0.6932	3963
macro-avg	0.7560	0.6929	0.7059	3963
Accuracy	0.6932			

Table 4.54 shows the classification details of AdaBoost classifier. As seen from the table, AdaBoost is a poor classifier according to the *f*-measure values and 69% of test documents correctly classified by AdaBoost. The confusion matrix can be seen from Figure 4.56 below.

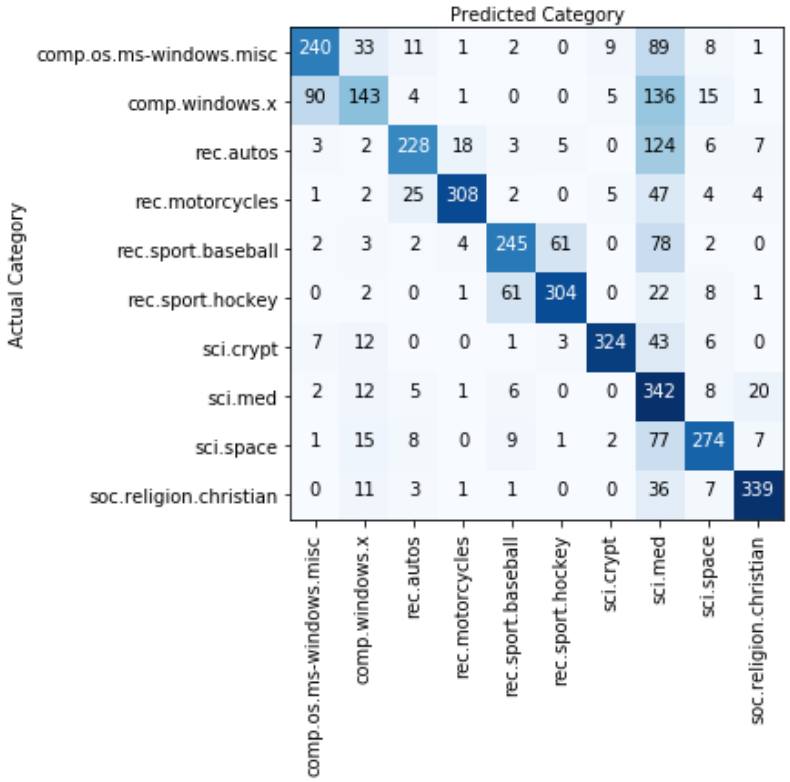

Figure 4.56: CM of AdaBoost on 20 Newsgroups Dataset

4.4.1.2 Using DF Weighting Scheme

In this section, the classification results are presented when the DF weighting scheme used with two different classification algorithms.

Table 4.55: CR of MNB with DF Weighting

	Multinomial Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.6088	0.6320	0.6202	394
comp.windows.x	0.6465	0.7038	0.6739	395
rec.autos	0.7065	0.6869	0.6965	396
rec.motorcycles	0.7246	0.8065	0.7634	398
rec.sport.baseball	0.8061	0.7330	0.7678	397
rec.sport.hockey	0.8658	0.8571	0.8615	399
sci.crypt	0.8170	0.8232	0.8201	396
sci.med	0.7609	0.5707	0.6522	396
sci.space	0.7989	0.7462	0.7717	394
soc.religion.christian	0.7773	0.9296	0.8467	398
micro-avg	0.7492	0.7492	0.7492	3963
macro-avg	0.7513	0.7489	0.7474	3963
Accuracy	0.7492			

Table 4.55 above shows the classification details of MNB classifier with DF term weighting scheme. As seen from the table, 75% of test documents correctly placed into corresponding categories. When it is compared with TF-IDF weighting, using DF alone negatively affects the classification process.

Table 4.56: CR of SVM (Linear Kernel) with DF Weighting

	SVM (Linear Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.5236	0.5914	0.5554	394
comp.windows.x	0.5044	0.5823	0.5405	395
rec.autos	0.4896	0.5934	0.5365	396
rec.motorcycles	0.7318	0.7060	0.7187	398
rec.sport.baseball	0.5991	0.6549	0.6258	397
rec.sport.hockey	0.7690	0.6842	0.7241	399
sci.crypt	0.7790	0.6944	0.7343	396
sci.med	0.5988	0.4975	0.5434	396
sci.space	0.6627	0.5584	0.6061	394
soc.religion.christian	0.8025	0.7965	0.7995	398
micro-avg	0.6361	0.6361	0.6361	3963
macro-avg	0.6460	0.6359	0.6384	3963
Accuracy	0.6362			

Table 4.56 above gives detailed information on the performance of linear SVM when it used with DF weighting. As seen from the table, the success rate of classification process decreased from 93% to 64% with DF term weighting scheme.

4.4.1.3 Using Probability-based Term Weighting

In this section, the results of probability-based term weighting approach are presented with MNB and SVM (Linear Kernel) classifiers. The tables 4.87 and 4.88 below show the classification results, and as seen from the tables, probabilistic term weighting approach is not a suitable method for categorizing the balanced datasets.

Table 4.57: CR of MNB with Probability-based Term Weighting

	Multinomial Naive Bayes Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.0000	0.0000	0.0000	394
comp.windows.x	0.0000	0.0000	0.0000	395
rec.autos	0.0000	0.0000	0.0000	396
rec.motorcycles	0.0000	0.0000	0.0000	398
rec.sport.baseball	0.0000	0.0000	0.0000	397
rec.sport.hockey	0.1007	1.0000	0.1829	399
sci.crypt	0.0000	0.0000	0.0000	396
sci.med	0.0000	0.0000	0.0000	396
sci.space	0.0000	0.0000	0.0000	394
soc.religion.christian	0.0000	0.0000	0.0000	398
micro-avg	0.1007	0.1007	0.1007	3963
macro-avg	0.0101	0.1000	0.0183	3963
Accuracy	0.1007			

Table 4.57 shows the performance of MNB classifier when the probability-based term weighting scheme is used.

Table 4.58: CR of SVM (Linear Kernel) with Probability-based Term Weighting

	SVM (Linear Kernel) Classifier			
	precision	**recall**	***f*-measure**	**Support**
comp.os.ms-windows.misc	0.0000	0.0000	0.0000	394
comp.windows.x	0.0000	0.0000	0.0000	395
rec.autos	0.0000	0.0000	0.0000	396
rec.motorcycles	0.0000	0.0000	0.0000	398
rec.sport.baseball	0.0000	0.0000	0.0000	397
rec.sport.hockey	0.0000	0.0000	0.0000	399
sci.crypt	0.0000	0.0000	0.0000	396
sci.med	0.0000	0.0000	0.0000	396
sci.space	0.0000	0.0000	0.0000	394
soc.religion.christian	0.1004	1.0000	0.1825	398
micro-avg	0.1004	0.1004	0.1004	3963
macro-avg	0.0100	0.1000	0.0183	3963
Accuracy	0.1004			

Table 4.58 above shows the classification result for linear SVM when the probability-based term weighting approach is used.

4.4.2 Performance of Feature Selection Methods

In this section, we provide the experimental results of assessing the performance of the feature selection metrics with two different classification algorithms for 20 Newsgroups dataset. The aim is to investigate the effectiveness of feature selection metrics and their achievements with different classifiers.

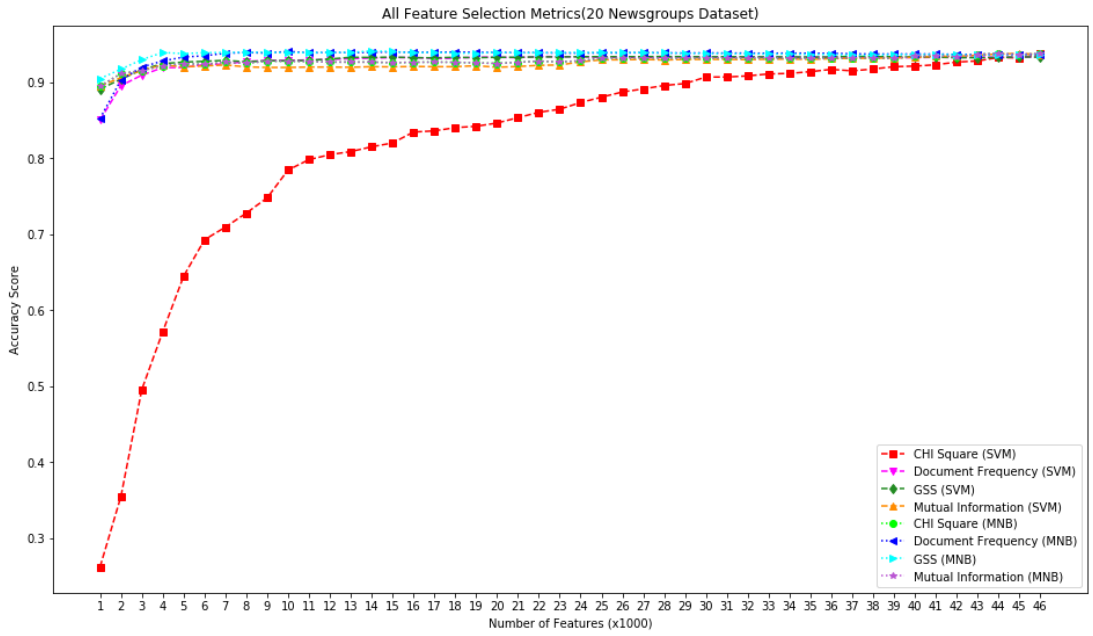

Figure 4.57: Performance of Feature Selection Metrics on 20 Newsgroups Dataset

Figure 4.57 above shows that the effectiveness of feature selection algorithms. As seen from the figure, all feature selection metrics were achieved the good performance especially GSS, MI and DF. When the results of the different dataset are compared, it is seen that the feature selection methods give better results on the balanced datasets.

4.5 Discussion

In this section, a study on the effectiveness of the four feature selection metrics and performance of classification algorithms is presented.

In our thesis, three different data sets used as highly skewed, skewed, and balanced. According to the macro and micro *f*-measure values obtained by the experiments performed with many classification algorithms, SVM and MLP have the highest success rate in highly skewed datasets. According to micro-averaged *f*-measure values, RF, kNN, DT, MNB, AdaBoost showed the highest success after the mentioned methods, while GNB is the worst classification method. When the macro-averaged *f*-measure values are taken into consideration, this order is changed to kNN, DT, RF, GNB, MNB and it is seen that the worst-performing classifier is AdaBoost.

In the skewed datasets, SVM and MLP also achieved the highest performance based on micro and macro-averaged *f*-measure values. After SVM and MLP according to micro-averaged criteria, the best results were obtained with AdaBoost, RF, MNB, DT, GNB and kNN, respectively.

In the experiments conducted with the dataset that has a balanced document distribution between the classes, it is found that the methods that give the best classification performance according to micro and macro f-measure values were MLP and MNB. Then it is seen that the most successful methods were SVM, kNN, GNB, RF, DT, and AdaBoost. Besides, in the dataset, which has a balanced distribution, it is also seen that the classification methods, according to macro and micro-average metrics, are all in the same success rate.

Precision, recall, *f*-measure, and accuracy metrics were used to analyze and interpret the results of our experiments. Although precision and recall metrics are essential in evaluating the success of classification methods, an inversely proportional relationship observed between these results. Therefore, in order to construct a balanced text classification system and to evaluate the performance more efficiently, it has been found more accurate to consider the harmonic average of these two metrics which is called, *f*-measure.

According to the results, it is observed that SVM is the most successful method in the classification of text data, MLP showed a remarkable performance, and sample-based classifiers are successful. The results of the study of Sebastiani F. [12] and the consequences of our experiments on the performance of classification algorithms coincide.

In the experiments conducted on the effect of term weighting methods on classification performance, the use of TF alone yielded good results, whereas the use of DF provided lower performance than TF. By combining these two methods, the best results obtained by TF-IDF weighting method. Besides, in highly skewed datasets, the probabilistic weighting method proposed by Liu, Y., Loh, H. T., & Sun, A. [34], positively improves the classification performance, while the use of this method in balanced datasets has been shown to have a negative effect.

As a result of the experiments conducted to measure the performance of feature selection methods, it is seen that the document distribution of datasets has a direct effect on feature selection. In the experiments, we performed three different datasets, and it is observed that the feature selection methods that provide the best performance

are DF and GSS. Besides, since MI scores highly for terms with high conditional probability, it is seen that the documents of dominant classes increase in weight. As a consequence, it has been observed that as the number of selected terms decreases, the classification performance decreases. According to our observations, the CHI method performs better than the MI method in the skewed datasets, it shows the same behavior as the MI method and performs poorly with the small subset of the feature set. In addition, it is found that all methods except CHI were successful with balanced datasets.

To sum up, it has been observed that SVMs are the best in text classification and do not even require operations such as feature selection, and in some cases, feature selection harms performance. It has been found that feature selection methods work well in datasets with balanced document distribution, also reduces computational complexity, and reduces the classification time and it makes possible the categorization process without loss of information by using fewer features. Even for the skewed dataset, DF and GSS were found to be more efficient than other methods.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis, a comparative study of statistical models for feature selection methods in text categorization presented. Firstly, we analyzed the performance of various learning machines on three different datasets with different levels of skewness by using different term weighting and preprocessing methods and then evaluated the effectiveness of CHI, MI, GSS and DF in order to choose most informative and discriminative features for the text categorization. Results are obtained by increasing the number of features by 1000 in each iteration in the range of 1000 to the total number of features. As a result, it is observed that GSS and DF methods are the most efficient methods for text categorization. It is also concluded that classifiers such as SVM and MLP are the most successful methods for text data.

5.2 Future Work

In this thesis, terms are weighted with TF-IDF, and then feature selection methods are applied. As a future work, the effect of the combination of term weighting and various feature selection metrics on classification performance will be studied.

In our study, we weighted each term individually and did not focus on the relationship between words. As a result, we have not examined the effect of the coexistence of words on determining the category of a document. For this reason, we plan to build a more effective model in the future with a weighting approach based on the sentences

and their positions in the document. We will also examine the effect of verbs, nouns, etc. on the classification performance by tagging words with part-of-speech tagging module.

As it is known, each text has an author, and each author has a different accumulation in terms of vocabulary. Because of the BoW model used in our study, each of the synonyms is considered as a distinct feature. In the future, we will try to build a more efficient classification model by changing the vocabulary of the corpus synthetically with word embedding using tools like WordNet and Word2Vec.

Finally, we focused on the single-label classification approach instead of multi-label. We plan to focus on multi-label classification techniques and related feature selection methods in the future.

REFERENCES

[1] Khan, A., Bahurdin, B. B., & Khan, K. (2011). An Overview of E-Documents Classification. In *2009 International Conference on Machine Learning and Computing IPCSIT* (Vol. 3).

[2] Pong, J. Y. H., Kwok, R. C. W., Lau, R. Y. K., Hao, J. X., & Wong, P. C. C. (2008). A comparative study of two automatic document classification methods in a library setting. *Journal of Information Science*, *34*(2), 213-230.

[3] Ko, Y., Park, J., & Seo, J. (2002, August). Automatic text categorization using the importance of sentences. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1* (pp. 1-7). Association for Computational Linguistics.

[4] Kalra, V., & Aggarwal, R. *Importance of Text Data Preprocessing & Implementation in RapidMiner.*

[5] Harish, B. S., & Revanasiddappa, M. B. (2017). A comprehensive survey on various feature selection methods to categorize text documents. *International Journal of Computer Applications*, *164*(8), 1-7.

[6] Yang, Y., & Pedersen, J. O. (1997, July). A comparative study on feature selection in text categorization. In *Icml* (Vol. 97, No. 412-420, p. 35).

[7] McHugh, M. L. (2013). The chi-square test of independence. *Biochemia medica: Biochemia medica*, *23*(2), 143-149.

[8] Church, K. W., & Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, *16*(1), 22-29.

[9] Uchyigit, G., & Ma, M. Y. (Eds.). (2008). *Personalization techniques and recommender systems* (Vol. 70). World Scientific.

[10] Dave, K. (2011). *Study of feature selection algorithms for text-categorization.*

[11] Chen, J., Huang, H., Tian, S., & Qu, Y. (2009). Feature selection for text classification with Naïve Bayes. *Expert Systems with Applications*, *36*(3), 5432-5435.

[12] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, *34*(1), 1-47.

[13] Cardoso-Cachopo, A., & Oliveira, A. L. (2006). Empirical evaluation of centroid-based models for single-label text categorization. *INSEC-ID Technical Report*, *7*, 2006.

[14] Lewis, D. (1997). Reuters-21578 text categorization test collection. *Distribution 1.0, AT&T Labs-Research*.

[15] The 4 Universities Dataset, *CMU Text Learning Group*, 1998, www.cs.cmu.edu/afs/cd/project/theo-20/www/data/.

[16] Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995* (pp. 331-339). Morgan Kaufmann.

[17] Hayes, P. J., & Weinstein, S. P. (1990, May). CONSTRUE/TIS: A System for Content-Based Indexing of a Database of News Stories. In *IAAI* (Vol. 90, pp. 49-64).

[18] Stamatatos, E., Fakotakis, N., & Kokkinakis, G. (2000). Automatic text categorization in terms of genre and author. *Computational linguistics*, *26*(4), 471-495.

[19] Goudjil, M., Koudil, M., Hammami, N., Bedda, M., & Alruily, M. (2013, June). Arabic text categorization using SVM active learning technique: An overview. In *2013 World Congress on Computer and Information Technology (WCCIT)* (pp. 1-2). IEEE.

[20] Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998, July). A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop* (Vol. 62, pp. 98-105).

[21] Srividhya, V., & Anitha, R. (2010). Evaluating preprocessing techniques in text categorization. *International journal of computer science and application*, *47*(11), 49-51.

[22] Leopold, E., & Kindermann, J. (2002). Text categorization with support vector machines. How to represent texts in input space?. *Machine Learning*, *46*(1-3), 423-444.

[23] Albishre, K., Albathan, M., & Li, Y. (2015, December). Effective 20 newsgroups dataset cleaning. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)* (Vol. 3, pp. 98-101). IEEE.

[24] Kadhim, A. I., Cheah, Y. N., & Ahamed, N. H. (2014, December). Text document preprocessing and dimension reduction techniques for text document clustering. In *2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology* (pp. 69-73). IEEE.

[25] Chandrasekar, P., & Qian, K. (2016, June). The Impact of Data Preprocessing on the Performance of a Naive Bayes Classifier. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* (Vol. 2, pp. 618-619). IEEE.

[26] Isa, D., Lee, L. H., Kallimani, V. P., & Rajkumar, R. (2008). Text document preprocessing with the Bayes formula for classification using the support vector machine. *IEEE Transactions on Knowledge and Data engineering*, *20*(9), 1264-1272.

[27] Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613-620.

[28] Zhang, Y., Jin, R., & Zhou, Z. H. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1-4), 43-52.

[29] Liu, M., & Yang, J. (2012). An improvement of TFIDF weighting in text categorization. *International proceedings of computer science and information technology*, 44-47.

[30] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, *24*(5), 513-523.

[31] Ishii, N., Murai, T., Yamada, T., & Bao, Y. (2006, July). Text classification by combining grouping, LSA and kNN. In *5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse (ICIS-COMSAR'06)* (pp. 148-154). IEEE.

[32] Liu, C. Z., Sheng, Y. X., Wei, Z. Q., & Yang, Y. Q. (2018, August). Research of Text Classification Based on Improved TF-IDF Algorithm. In *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)* (pp. 218-222). IEEE.

[33] Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

[34] Liu, Y., Loh, H. T., & Sun, A. (2009). Imbalanced text classification: A term weighting approach. *Expert systems with Applications*, *36*(1), 690-701.

[35] Özgür, A., Özgür, L., & Güngör, T. (2005, October). Text categorization with class-based and corpus-based keyword selection. In *International Symposium on Computer and Information Sciences* (pp. 606-615). Springer, Berlin, Heidelberg.

[36] Ladha, L., & Deepa, T. (2011). Feature selection methods and algorithms. *International journal on computer science and engineering*, *3*(5), 1787-1797.

[37] Rogati, M., & Yang, Y. (2002, November). High-performing feature selection for text classification. In *Proceedings of the eleventh international conference on Information and knowledge management* (pp. 659-661). ACM.

[38] Hawashin, B., Mansour, A., & Aljawarneh, S. (2013). An efficient feature selection method for Arabic text classification. *International journal of computer applications*, *83*(17).

[39] Saleh, S. N., & El-Sonbaty, Y. (2007, November). A feature selection algorithm with redundancy reduction for text classification. In *2007 22nd international symposium on computer and information sciences* (pp. 1-6). IEEE.

[40] Bakus, J., & Kamel, M. S. (2006). Higher order feature selection for text classification. *Knowledge and Information Systems*, *9*(4), 468-491.

[41] Zhang, H., Ren, Y. G., & Yang, X. (2013, November). Research on text feature selection algorithm based on information gain and feature relation tree. In *2013 10th Web Information System and Application Conference* (pp. 446-449). IEEE.

[42] Joachims, T. (1998, April). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning* (pp. 137-142). Springer, Berlin, Heidelberg.

[43] Yang, Y., & Chute, C. G. (1994). An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems (TOIS)*, *12*(3), 252-277.

[44] Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In *Mining text data* (pp. 163-222). Springer, Boston, MA.

[45] Niharika, S., Latha, V. S., & Lavanya, D. R. (2012). A survey on text categorization. *International Journal of Computer Trends and Technology*, *3*(1), 39-45.

[46] Loper, E., & Bird, S. (2002). NLTK: the natural language toolkit. *arXiv preprint cs/0205028*.

[47] Salton, G. (1971). The SMART system. *Retrieval Results and Future Plans*.

[48] Kanadje, M. (2014, June). *Reuters-21578 Stopwords*. Retrieved from https://github.com/manishkanadje/reuters-21578/blob/master/stopwords.txt.

[49] Willett, P. (2006). The Porter stemming algorithm: then and now. *Program*, *40*(3), 219-223.

[50] Zobel, J., & Moffat, A. (1998, April). Exploring the similarity space. In *Acm Sigir Forum* (Vol. 32, No. 1, pp. 18-34). ACM.

[51] Li, Y., Hsu, D. F., & Chung, S. M. (2009, November). Combining multiple feature selection methods for text categorization by using rank-score characteristics. In *2009 21st IEEE International Conference on Tools with Artificial Intelligence* (pp. 508-517). IEEE.

[52] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, *12*(Oct), 2825-2830.

[53] Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[54] Jones, E., Oliphant, T., & Peterson, P. (2001). SciPy: Open source scientific tools for Python.

[55] *The 20 Newsgroups Dataset*, 2008, http://qwone.com/~jason/20Newsgroups/.

[56] Sadr H., Atani R. E., Yamaghani M. R. (2014, May). The Significance of Normalization Factor of Documents to Enhance the Quality of Search in

Information Retrieval Systems. *International Journal of Computer Science and Network Solutions 2 (5), 91-97.*

[57] Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

[58] Frank, E., & Bouckaert, R. R. (2006, September). Naive bayes for text classification with unbalanced classes. In *European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 503-510). Springer, Berlin, Heidelberg.

[59] Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G. (2004, December). Multinomial naive bayes for text categorization revisited. In *Australasian Joint Conference on Artificial Intelligence* (pp. 488-499). Springer, Berlin, Heidelberg.

[60] Murphy, K. P. (2006). Naive bayes classifiers. *University of British Columbia*, *18*, 60.

[62] Rokach, L., & Maimon, O. Z. (2008). *Data mining with decision trees: theory and applications* (Vol. 69). World scientific.

[63] Du, W., & Zhan, Z. (2002, December). Building decision tree classifier on private data. In *Proceedings of the IEEE international conference on Privacy, security and data mining-Volume 14* (pp. 1-8). Australian Computer Society, Inc..

[64] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

[65] Schapire, R. E. (2013). Explaining adaboost. In *Empirical inference* (pp. 37-52). Springer, Berlin, Heidelberg.

[66] Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron) a review of applications in the atmospheric sciences. *Atmospheric environment*, *32*(14-15), 2627-2636.

[67] Apté, C., Damerau, F., & Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, *12*(3), 233-251.

[68] Manning, C., Raghavan, P., & Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, *16*(1), 100-103.